DISEÑO Y FABRICACIÓN DE LA ELECTRÓNICA PARA UN SISTEMA DE TRANSFERENCIA INALÁMBRICA DE ENERGÍA BASADO EN METAMATERIALES

RUBÉN SÁNCHEZ MÍNGUEZ

DIRECTORES:
José Sánchez-Dehesa Moreno-Cid
Francisco Cervera Moreno

Un trabajo final de máster elaborado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València

Máster Universitario en Ingeniería de Sistemas Electrónicos (MUISE)
Julio 2017

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
GFO Grupo de Fenomenos Ondulatorios

Rubén Sánchez Mínguez: *Diseño y fabricación de la electrónica para un sistema de transferencia inalámbrica de energía basado en metamateriales*, Máster Universitario en Ingeniería de Sistemas Electrónicos (MUISE) , Julio 2017

DIRECTOR:
José Sánchez-Dehesa Moreno-Cid

Dedicado a la memoria de Vicente Villaplana Valensa.

1953 – 2017

ABSTRACT

The purpose of this project consists of the characterization of a wireless power transfer system using metal-dielectric resonators based on metamaterials. Moreover, the project also includes the design and implementation of the transceiver stage that will provide the power to the system.

RESUM

El propòsit d'aquest projecte consisteix en la caracterització d'un sistema de transferència de potència sense fils utilitzant resonadors metalodielèctrics basats en metamaterials. A més, també inclou el disseny i la implementació de l'etapa de transmissió que proporcionarà potència al sistema.

RESUMEN

El propósito de este proyecto consiste en la caracterización de un sistema de transferencia inalámbrica de potencia utilizando resonadores metalodieléctricos basados en metamateriales. Además, también incluye el diseño y la implementación de la etapa de transmisión que proporcionará potencia al sistema.

AGRADECIMIENTOS

Mis más sinceros agradecimientos para José Sánchez-Dehesa y Francisco Cervera, directores del proyecto, sin los cuales no hubiese sido posible su realización. Además, agradezco la hospitalidad del Grupo de Fenómenos Ondulatorios (GFO) durante el desarrollo de este proyecto. También quiero agradecer a Víctor Manuel García por sus ideas y su constante ayuda a lo largo de todo el proyecto. Finalmente, agradecer a mi familia y amigos por su apoyo incondicional durante el transcurso del proyecto, con especial mención a Gema.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Parte I

INTRODUCCIÓN

1 INTRODUCCIÓN

Durante los últimos años, la transferencia inalámbrica de potencia ha sido un tema candente debido al gran crecimiento de dispositivos inalámbricos de comunicación. Un aspecto que ha cambiado poco durante los últimos tiempos es la forma en la que estos dispositivos se alimentan o se recargan, siendo las aproximaciones más extendidas aquellas que hacen uso de cables. Sin embargo, esta implementación supone una restricción en aspectos como la movilidad y dado el nivel de integración que estos dispositivos está alcanzando, cada vez los conectores son una parte relativa más grande de todo el producto. Por tanto, la transferencia inalámbrica de potencia solucionaría estos problemas, dotando de una mayor versatilidad y nuevos modos de uso de productos electrónicos como teléfonos móviles, ordenadores y sensores.

1.1 TRANSFERENCIA INALÁMBRICA DE ENERGÍA

La idea de transferir energía sin cables ha sido contemplada desde hace años, tal y como puede observarse en los trabajos de Nikola Tesla [1] de principios del siglo 20. Para una revisión completa de la tecnología consultar [2].

Existen varios métodos para ello, como son las técnicas de campo lejano o las de campo cercano. Las técnicas de campo lejano consisten en propagar ondas electromagnéticas del mismo modo que se hace en comunicaciones, con el objetivo último de transmitir potencia en lugar de transmitir información. De esta manera, se pueden cubrir grandes distancias siempre que se utilicen antenas de alta directividad. De lo contrario, la eficiencia del sistema decrece en un factor $1/r^2$ con la distancia debido a la transmisión omnidireccional de la señal. Uno de los problemas de estas implementaciones reside en el hecho de que al utilizar antenas de alta directividad con campos radiados, es necesario que haya línea de visión (LoS) continua y constante para que conseguir una transferencia de energía efectiva.

Por otra parte, existen las técnicas de campo cercano o de acoplamiento inductivo como las que utiliza RFID y que se emplean en dispositivos como teléfonos móviles o cepillos de dientes. Sin embargo, su rango de operación es del orden de decenas de centímetros con una eficiencia del 1 o 2 % [3]. Productos para la carga inalámbrica de dispositivos electrónicos portátiles utilizando esta tecnología ya han sido comercializados y la mayoría lo hace implementando el estándar de interfaz *Qi* desarrollado por el *Wireless Power Consortium* [4].

Como alternativa a estas técnicas, existe la resonancia magnética acoplada. Dos objetos con la misma frecuencia de resonancia se transfieren energía de manera eficiente, mientras que la energía transferida a otros objetos que no tienen la misma frecuencia de resonancia se mantiene baja. Por otra parte, la transferencia de potencia de medio alcance mediante resonancia magnética acoplada suele tener un comportamiento más parecido al omnidireccional, por lo que el posicionamiento de los resonadores no es tan crítico como en el caso de los campos radiados mediante antenas de alta directividad.

El sistema típico de transferencia inalámbrica de potencia mediante resonancia magnética acoplada utiliza cuatro bobinas, tal y como se muestra en la figura 1 [5]. Las bobinas A y B son de una única espira, mientras que las bobinas Tx y Rx son bobinas multivuelta. Para la transferencia de energía, se aplica potencia en la bobina A, la cual excita la bobina Tx que almacena la energía del mismo modo que ocurre en un tanque LC. Debido a la resonancia entre la bobina Tx y Rx, esta energía almacenada se transfiere al otro extremo. Finalmente, de la misma manera que ocurre en el circuito de excitación, se transfiere la energía de la bobina Rx a la carga. La interacción entre la bobina A y Tx y entre la bobina Rx y B se puede interpretar como el mismo mecanismo que existe en un transformador.

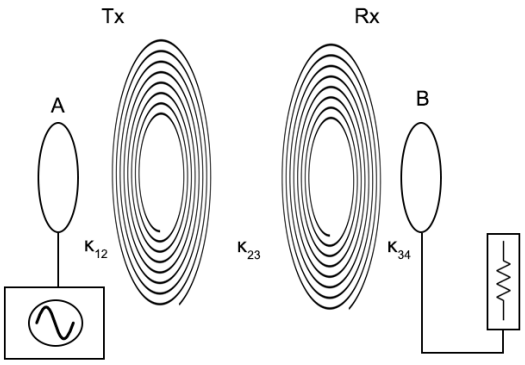

Figura 1: Sistema de transferencia inalámbrica de potencia mediante bobinas acopladas magnéticamente

El modelo circuital que se ajusta a este sistema es el que aparece en la figura 2, en el que cada bobina se representa mediante un tanque RLC. Las bobinas de una única vuelta se pueden modelar como un inductor de valor L_1, cuya resistencia parásita es R_{p1} y al que se añade la capacidad C_1 para que resuene a la frecuencia de trabajo. Lo mismo ocurre en la bobina B, donde en lugar de tener conectado el generador está la carga que se quiere alimentar. Las bobinas Tx y Rx también se modelan mediante un tanque RLC, cuya capacidad viene determinada por su geometría y R_{p2} y R_{p3} representan la resistencia parásita que modela las pérdidas en las bobinas Tx y Rx respectivamente. Los inductores L_1 y L_2 están acoplados mediante el coeficiente de acoplamiento κ_{23}.

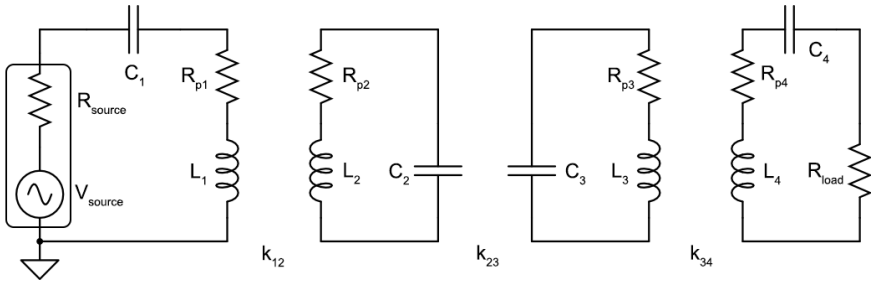

Figura 2: Modelo equivalente del sistema de transferencia inalámbrica de energía

Además de poder representar un sistema de resonancia magnética acoplada a través de su modelo circuital, es posible caracterizar un sistema de transferencia inalámbrica de potencia de este tipo a través de la matriz de dispersión o parámetros S. Aunque las matrices de parámetros Z son útiles para modelar un sistema de transferencia inalámbrica de potencia, la mayoría trabajan en un rango de frecuencias alrededor de las decenas o centenares de MHz. Es por ello que el uso de los parámetros S es adecuado para este tipo de aplicaciones. El sistema completo se puede representar de forma genérica como una red de dos puertos en cuya entrada se conecta una fuente de alimentación y a la salida se conecta una carga tal y como se muestra en la figura 3. En ella se identifican las ondas de potencia incidentes a_1 y a_2 y las ondas de potencia reflejadas b_1 y b_2, que se define según en Ec. 1 donde Z_0 es la impedancia característica de la línea de transmisión utilizada.

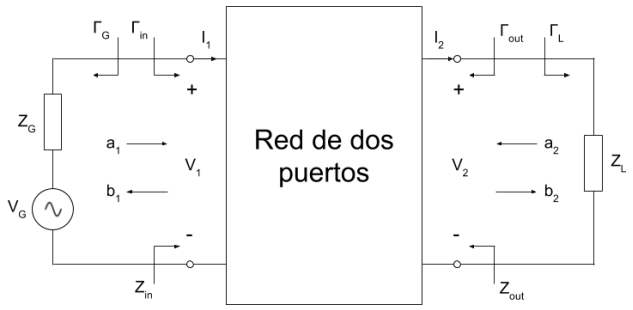

Figura 3: Red de dos puertos conectada a una fuente de alimentación y una carga

$$\begin{aligned} a_1 &= \frac{V_1 + Z_0 I_1}{2\sqrt{Z_0}}, \quad a_2 = \frac{V_2 - Z_0 I_2}{2\sqrt{Z_0}}, \\ b_1 &= \frac{V_1 - Z_0 I_1}{2\sqrt{Z_0}}, \quad b_2 = \frac{V_2 + Z_0 I_2}{2\sqrt{Z_0}}. \end{aligned} \tag{1}$$

La matriz de dispersión relaciona las ondas incidentes en el puerto 1 con las ondas salientes del puerto 2.

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{2}$$

A partir de Ec. 1 se puede demostrar que $b_1 = \Gamma_{in} a_1$ y que $a_2 = \Gamma_L b_2$ donde Γ_{in} y Γ_L son los coeficientes de reflexión de la entrada y de la carga que se corresponde a las expresiones de la Ec. 3. Por simetría, también se pueden definir los coeficientes de reflexión del generador (Γ_G) y de la salida (Γ_{out}) de acuerdo con la Ec. 4.

$$\begin{aligned} \Gamma_{in} &= \frac{Z_{in} - Z_0}{Z_{in} + Z_0}, \\ \Gamma_L &= \frac{Z_L - Z_0}{Z_L + Z_0}. \end{aligned} \tag{3}$$

$$\begin{aligned} \Gamma_G &= \frac{Z_G - Z_0}{Z_G + Z_0}, \\ \Gamma_{out} &= \frac{Z_{out} - Z_0}{Z_{out} + Z_0}. \end{aligned} \tag{4}$$

La eficiencia del sistema se puede describir a través de la ganancia operativa del sistema G_p [6] que corresponde al cociente de la potencia que sale de la red y la potencia que entra en la red. De esta manera no se tiene en cuenta la potencia reflejada en la entrada de la red debido a desadaptaciones y se calcula la eficiencia neta de transmisión. Utilizando esta definición de eficiencia es posible comparar diferentes sistemas de transferencia inalámbrica de potencia ya que independiza las reflexiones de la entrada de la eficiencia del sistema. De este modo, solo se contabiliza la cantidad de energía que habiendo entrado en la red es capaz de entregarse a la carga. Esta definición corresponde a la expresión de la Ec. 5, donde P_{in} y P_L se calculan de acuerdo con Ec. 6.

$$\eta = G_p = \frac{P_L}{P_{in}} \tag{5}$$

$$\begin{aligned} P_{in} &= \frac{1}{2}\Re(V_1^* I_1) = \frac{|V_G|^2}{8Z_0} \frac{\left(1 - |\Gamma_{in}|^2\right) |1 - \Gamma_G|^2}{|1 - \Gamma_{in}\Gamma_G|^2}, \\ P_L &= \frac{1}{2}\Re(V_2^* I_2) = \frac{|V_G|^2}{8Z_0} \frac{\left(1 - |\Gamma_L|^2\right) |1 - \Gamma_G|^2 |S_{21}|^2}{|(1 - S_{11}\Gamma_G)(1 - S_{22}\Gamma_L) - S_{12}S_{21}\Gamma_G\Gamma_L|^2} \end{aligned} \tag{6}$$

De esta manera, es posible llegar a la definición de eficiencia que se muestra en la Ec. 7.

$$\eta = \frac{|S_{21}|^2 \left(1 - |\Gamma_L|^2\right)}{\left(1 - |\Gamma_{in}|^2\right) |1 - S_{22}\Gamma_L|^2} \tag{7}$$

Sin embargo, en un sistema en el que la carga esté adaptada a la impedancia característica Z_0 de modo que $\Gamma_L = 0$, se obtiene la expresión simplificada de la Ec. 8.

$$\eta = \frac{|S_{21}|^2}{1 - |S_{11}|^2} \tag{8}$$

1.2 METAMATERIALES

Un metamaterial es un material artificial que tiene propiedades electromagnéticas que no se encuentran en la naturaleza. Las ecuaciones de Maxwell proporcionan unas reglas que describen cómo los campos electromagnéticos interactúan con la materia de acuerdo con las expresiones de la Ec. 9, en donde la permeabilidad magnética μ y la permitividad eléctrica ϵ son propiedades electromagnéticas intrínsecas del material.

$$\begin{aligned} \nabla \times \vec{E} &= -j\omega\mu\vec{H}, \\ \nabla \times \vec{H} &= j\omega\epsilon\vec{E} \end{aligned} \tag{9}$$

De manera natural, la mayoría de materiales tienen una ϵ y μ positiva. Sin embargo, estos nuevos materiales fabricados artificialmente pueden tener permitividades eléctricas y permeabilidades magnéticas negativas [6]. De este modo, si se cambian los signos de ϵ y μ en las ecuaciones de Maxwell, se obtienen una tripla a izquierdas del campo eléctrico **E**, el campo magnético **H** y el vector de fase **k** que en caso de los materiales normales es a derechas. Esto tiene implicaciones en el vector de Poynting **S**, ya que la fase de la onda progresa en sentido contrario a la dirección de propagación de la energía [7].

1.3 ANTECEDENTES DEL PROYECTO

El proyecto se basa en el trabajo previo de Ana Díaz-Rubio, Jorge Carbonell y José Sánchez-Dehesa en el que se realiza un estudio de estructuras con cristales fotónicos radiales para la transferencia inalámbrica de potencia [8]. A partir de la teoría de modos acoplados [9], de la que se puede extraer que cualquier tipo de resonador con un factor de calidad Q alto y que trabaje en régimen de alto acoplo

puede transmitir de manera eficaz energía [6], se ha estudiado una configuración cuya permeabilidad magnética μ y permitividad eléctrica ϵ varía en el eje radial de acuerdo con las expresiones de la Ec. 10 y que se muestran en la figura 4.

$$\begin{aligned} \mu_{\theta a}(r) &= \frac{35d}{r}; \quad \mu_{\theta b}(r) = \frac{20d}{r} \\ \epsilon_{za}(r) &= \frac{40d}{r}; \quad \epsilon_{zb}(r) = \frac{60d}{r} \end{aligned} \tag{10}$$

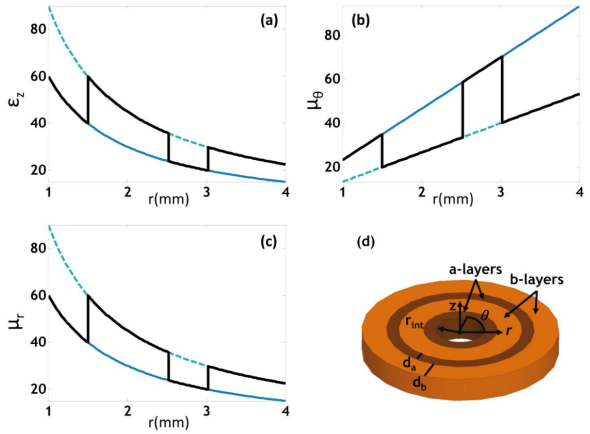

Figura 4: a) Perfil de la permitividad eléctrica ϵ_z. b) Perfil de la permeabilidad angular μ_θ. c) Perfil de permeabilidad radial μ_r. d) RPC de 4 capas.

Con este tipo de resonadores se consigue una eficiencia teórica de hasta el 83 % y una mejora en la figura de mérito (FOM) definida como el cociente del acoplamiento (κ) entre las pérdidas totales de los discos (Γ) respecto a resonadores cuyos parámetros constitutivos son el promedio de la estructura propuesta, tal y como puede observarse en la figura 5.

Esta estructura radial se ha simplificado para obtener una configuración que sea implementable manteniendo el mismo principio. Esta estructura final es la que se muestra en la figura 6 donde se puede observar que existe un soporte metálico de aluminio en forma de cruz y a la que se adhieren cuatro cerámicas. Los parámetros que definen un par de resonadores son los que aparecen en la figura, donde R es el radio externo del disco, r es el radio interno que existe entre el centro de la circunferencia y la cerámica, h es el grosor del disco y ϵ es la permitividad eléctrica de las cerámicas.

Esta estructura simplificada se puede interpretar del mismo modo que el modelo de la figura 2 de cuatro bobinas. De esta manera, los resonadores equivalen a las bobinas Tx y Rx, mientras que las espiras para inyectar y extraer la potencia son los conectores utilizados.

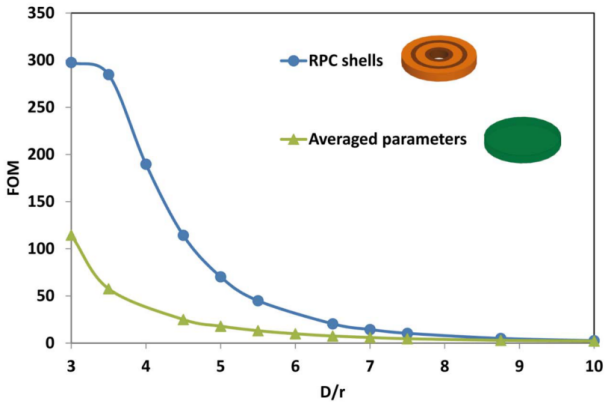

Figura 5: FOM $= \kappa/\Gamma$ de la estructura de cristales fotónicos radiales en comparación a un disco homogéneo en función de la distancia de separación (D) normalizada respecto al radio los discos (r).

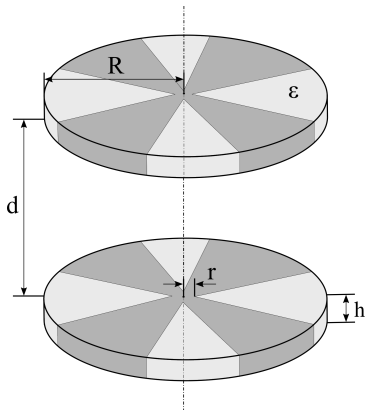

Figura 6: Características de los resonadores

1.4 MÉTODOS Y PROCEDIMIENTOS

El proyecto ha sido realizado en el Grupo de Fenómenos Ondulatorios (GFO) del Departamento de Ingeniería Electrónica (DIE) de la Universitat Politècnica de València (UPV) y está basado en el trabajo previo del grupo, partiendo del estudio teórico de la estructura y de las cerámicas fabricadas en el Instituto de Ciencia de Materiales de Madrid (ICMM-CSIC). El resto de contenidos han sido elaborados durante la realización del proyecto y forman parte de él.

Parte II

OBJETIVOS

2 OBJETIVOS

Los objetivos del proyecto se pueden dividir en dos partes. En primer lugar, se desea montar y caracterizar los resonadores a fin de estudiar su comportamiento frente a diferentes condiciones operativas como frecuencia o posición relativa entre resonadores. El segundo objetivo del proyecto es el de diseñar la electrónica de transmisión de manera compacta para que pueda servir a modo de demostrador tecnológico portátil.

2.1 MONTAJE Y CARACTERIZACIÓN DE LOS RESONADORES

Existen dos motivos por los que es necesario caracterizar los resonadores. El primero consiste en poder conocer cuáles son las características de cada una de las cerámicas. De esta manera, será posible realizar el posterior diseño de la electrónica de transmisión para aquellos resonadores que mejor comportamiento y eficiencia tengan.

Además, los datos extraídos de las mediciones se utilizarán como material de un artículo científico en el que se detallará las bases teóricas y los resultados experimientales de este tipo de estructuras metalodieléctricas para su uso en aplicaciones de transferencia inalámbrica de energía.

Se estudiará la variación de la eficiencia y las pérdidas de retorno en función de diferentes variables del sistema. Las variables más interesantes que se monitorizarán para cada uno de los resonadores son:

- Frecuencia: comportamiento del resonador en función de la frecuencia de la señal con la que es excitado.
- Distancia vertical: distancia mínima que existe entre las caras más próximas de los resonadores.
- Distancia horizontal: distancia entre los ejes normales a la superficie de cada resonador.
- Ángulo radial: ángulo que forman los dos resonadores respecto a una posición de alineación azimutal.
- Ángulo axial: ángulo que existe entre los ejes normales de la superficie de cada resonador.

En la figura 7 se muestra una representación gráfica del significado de cada una de las variables anteriores.

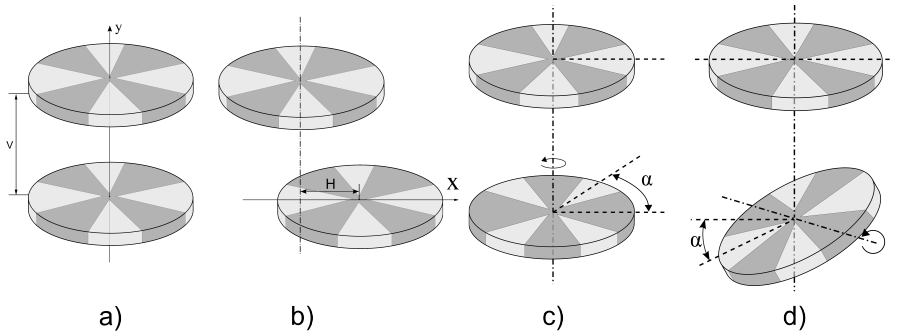

Figura 7: a) Desplazamiento vertical. b) Desplazamiento horizontal. c) Rotación radial. d) Rotación axial.

2.2 DISEÑO DE LA ELECTRÓNICA DE TRANSMISIÓN

Una vez caracterizados los resonadores se diseñará un transmisor portátil para el resonador cuyas características sean más favorables. De este modo, se dispondrá de una demostrador tecnológico de dimensiones reducidas que sea fácilmente transportable. Los objetivos marcados para el transmisor son conseguir transmitir tanta potencia como sea posible, fijando un orden de magnitud alrededor de 1 W. También de ser posible el ajuste de la frecuencia de salida para que el transmisor pueda sintonizarse a la frecuencia de resonancia de los resonadores. Por tanto, una variación precisa de la frecuencia será necesaria. De este modo, los objetivos iniciales son:

- Potencia: tan alta como sea posible, fijando una primera aproximación alrededor de 1W.
- Frecuencia: ajustable alrededor de la frecuencia de resonancia. Dependiendo del resonador elegido la frecuencia de la señal de salida será diferente.

El objetivo de potencia no es estricto ya que debido a la frecuencia de resonancia de los discos y la falta de soluciones comerciales de alta potencia para el rango de frecuencias de funcionamiento del sistema, no es posible conseguir unos valores de potencia más allá de los valores fijados inicialmente.

Parte III

METODOLOGÍA

3 METODOLOGÍA

El desarrollo del proyecto se divide en tres partes: fabricación de los resonadores, su caracterización y el diseño e implementación de los transmisores. En la caracterización se encuentran las medidas realizadas con el montaje manual y el montaje automático. Por último, la parte de implementación se divide en el diseño del transmisor para el resonador R3 y el transmisor para el resonador R4.

3.1 GESTIÓN DEL PROYECTO

El proyecto se ha desarrollado dentro del Grupo de Fenómenos Ondulatorios del Departamento de Ingeniería Electrónica. De esta manera, el seguimiento del proyecto ha sido continuo y constante en cada una de sus fases.

3.2 DISTRIBUCIÓN EN TAREAS

El plan de trabajo se puede descomponer en 4 bloques:

1. Fabricación de los resonadores: creación de los soportes metálicos y ensamblado de las cerámicas
2. Caracterización de los resonadores: estudio del comportamiento de los resonadores en función de la distancia de separación, el ángulo axial y el ángulo radial.
   - 2.1. Caracterización manual: cambio de las condiciones (separación, ángulo) de manera manual.
   - 2.2. Caracterización automática: cambio de condiciones operativas mediante etapas motorizadas y programadas a través de LabVIEW.
3. Procesado de datos: elaboración de gráficas y conclusiones a partir de las medidas tomadas.
4. Diseño de transmisores: diseño del transmisor para el resonador R3 y el transmisor para el resonador R4.

3.2.1 *Diagrama de Gantt*

	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		Fabricación del resonador R2	9 horas	lun 27/06/16	mar 28/06/16
2		Montaje de las cerámicas R2	5 horas	mar 28/06/16	mié 29/06/16
3		Caracterización R2	15 horas	mié 29/06/1	jue 30/06/1
4		Análisis de datos R2	6 horas	vie 01/07/16	vie 01/07/1
5		Fabricación del resonador R3	10 horas	vie 01/07/16	mar 05/07/16
6		Montaje de las cerámicas R3	5 horas	mar 05/07/16	mar 05/07/16
7		Caracterización R3	15 horas	mié 06/07/1	jue 07/07/1
8		Análisis de datos R3	6 horas	jue 07/07/16	vie 08/07/1
9		Diseño del transmisor R3	23 días	vie 08/07/16	lun 09/01/17
10		Montaje del transmisor R3	5 horas	lun 09/01/17	mar 10/01/17
11		Caracterización del transmisor R3	15 horas	mar 10/01/17	mié 11/01/17
12		Fabricación del resonador R4	15 horas	jue 12/01/17	vie 13/01/1
13		Montaje de las cerámicas R4	5 horas	vie 13/01/17	lun 16/01/17
14		Pruebas de potencia con montaje de laboratorio	3 días	lun 16/01/17	mié 18/01/17
15		Toma de medidas de potencia	3 días	mié 18/01/17	vie 20/01/1
16		Caracterización R4	15 horas	vie 20/01/17	mar 24/01/
17		Análisis de datos R4	5 horas	mar 24/01/1	mié 25/01/
18		Fabricación del resonador R5	15 horas	mié 25/01/17	vie 27/01/1
19		Montaje de las cerámicas R5	5 horas	vie 27/01/17	vie 27/01/1
20		Caracterización R5	15 horas	lun 30/01/17	mié 01/02/
21		Análisis de datos R5	5 horas	mié 01/02/1	mié 01/02/
22		Diseño del transmisor R4	40 días	mié 01/02/17	vie 10/03/1
23		Montaje del transmisor R4	5 horas	vie 10/03/17	lun 13/03/17
24		Caracterización del transmisor R4	15 horas	lun 13/03/17	mié 15/03/17
25		Fabricación del resonador R6	15 horas	mié 15/03/17	vie 17/03/1
26		Montaje de las cerámicas R6	5 horas	vie 17/03/17	lun 20/03/17
27		Caracterización R6	15 horas	lun 20/03/17	mié 22/03/
28		Análisis de datos R6	5 horas	mié 22/03/1	mié 22/03/
29		Fabricación del resonador R7	20 horas	jue 23/03/17	lun 27/03/17
30		Montaje de las cerámicas R7	5 horas	lun 27/03/17	mar 28/03/17
31		Caracterización R7	15 horas	mar 28/03/1	jue 30/03/1
32		Análisis de datos R7	5 horas	jue 30/03/17	jue 30/03/1
33		Montaje motorizado	15 días	vie 31/03/17	vie 28/04/1
34		Creación de los programas de LabVIEW	5 días	vie 28/04/17	jue 04/05/17
35		Toma de medidas bidimensionales	21 días	jue 04/05/17	mié 24/05/17
36		Creación de los script MATLAB para analizar los datos	3 días	mié 24/05/17	vie 26/05/1
37		Video de demostración	10 horas	vie 26/05/17	lun 29/05/17
38		Escribir memoria	25 días	mar 30/05/1	mié 21/06/

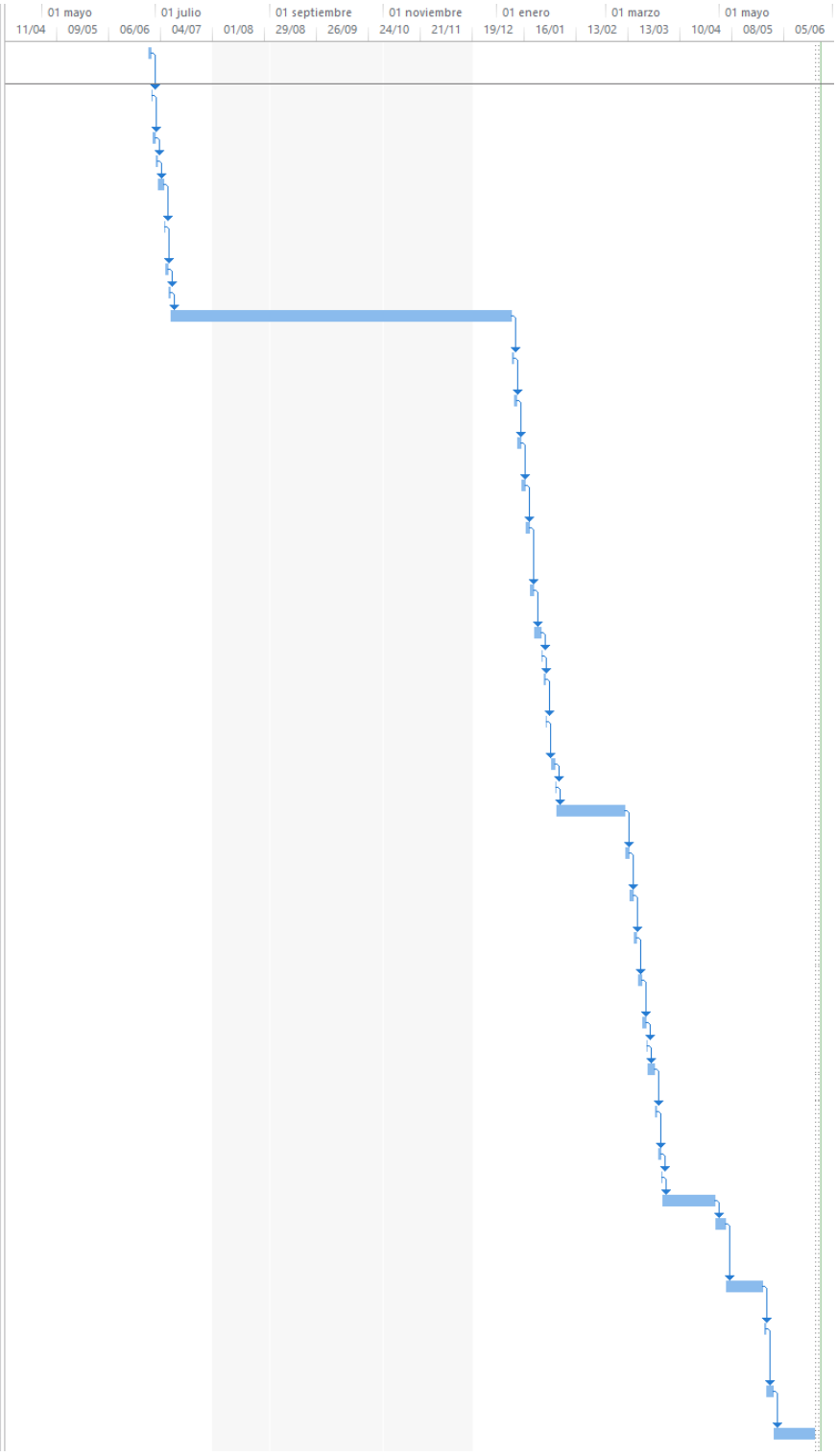

Figura 8: Diagrama de Gantt

3.3 INCIDENCIAS EN EL PLAN DE TRABAJO

Durante el desarrollo del proyecto han habido algunas incidencias que han obligado a modificar el plan de trabajo previsto. A pesar de que no han supuesto un gran retraso para su finalización, a continuación se detallan.

- Demora en la creación de montaje de la caracterización.
- Problemas técnicos con la instrumentación de medida y el montaje motorizado.
- Retraso en el diseño del transmisor.

Parte IV

DESARROLLO Y RESULTADOS DEL PROYECTO

4

DESARROLLO Y RESULTADOS DEL PROYECTO

4.1 FABRICACIÓN DE LOS DISCOS METALODIELÉCTRICOS

Los resonadores están formados por un soporte metálico y cuatro piezas cerámicas. Es por ello que es necesario la fabricación del soporte metálico en el que se adherirán las cerámicas.

4.1.1 *Fabricación del disco metálico*

Para la fabricación del soporte metálico se ha utilizado una fresadora de control numérico CC-F1210 de la marca Orpi de tres grados de libertad, es decir, capaz de moverse en la dirección X, Y y Z, con una precisión de 10 µm. Esta fresadora se configura a través de programas escritos en G-code (también conocido como RS-274) con el que se describen todos los movimientos que deben realizar los motores. El material del cual se extraen los soportes metálicos es aluminio.

La forma del soporte metálico y del disco resonador terminado se puede observar en la figura 9. Para la fabricación del soporte metálico se siguen tres pasos:

1. Rebaja de la plancha de aluminio: el programa se encarga de realizar un barrido por todo el prisma metálico para que el espesor del aluminio sea el mismo que el de las cerámicas.

2. Vaciado de los huecos de las cerámicas: el segundo paso para la fabricación del soporte metálico es el vaciado del metal para crear la estructura de soporte (ver figura 9.a) donde las cerámicas serán insertadas. Este proceso se repite cuatro veces en la misma plancha metálica para conseguir los huecos donde se introducirán las cerámicas.

3. Vaciado de la circunferencia externa: una vez realizado el espacio donde se insertarán las cerámicas, solo es necesario vaciar el aluminio alrededor de la cruz que forma para poder extraerla.

En total se han fabricado los discos para 6 resonadores distintos.

4.1.2 *Montaje de las cerámicas*

Para fijar la cerámica al disco metálico se utiliza una resina epoxi mezclada con plata para asegurar la continuidad eléctrica entre la interfaz del disco y la pared de las cerámicas. Dependiendo de la marca

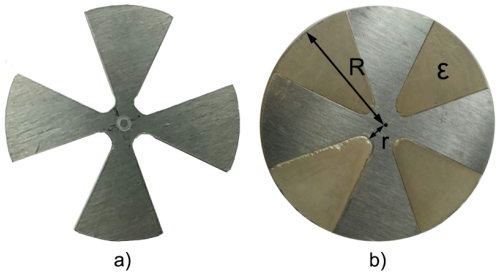

Figura 9: a) Soporte metálico sin cerámicas. b) Resonador terminado

concreta del poliepóxido, el tiempo de curado varía. En el desarrollo de este proyecto se han utilizado dos marcas distintas (RS [10] y MG Chemicals [11]) con tiempos de curado de 15 minutos a 95 °C para el caso de RS y 2 horas a 65 °C para MG Chemicals. Para ello se ha utilizado una estufa UF30 Plus del fabricante Memmert con la que se ha podido controlar térmicamente todo el proceso.

WPT	Radio exterior (mm)	Radio interior (mm)	Grosor (mm)	$\tan(\delta)$ (%)	ϵ	Composición
R2	30	5,2	5			
R3	32	5,6	7			
R4	33,225	5,11	10,51			
R5	31,325	5,07	6	0,4	641	BST 0.6 Sr
R6	32	5,075	5	0.6 (1Hz)	2376 (1 Hz)	BTO Inframat
R7	28,65	5.18	6,3	0.1	980 (1 Hz)	BST 0.6 Sr

Tabla 1: Resumen de los discos resonadores fabricados

4.2 CONECTORIZACIÓN

Una vez montados los resonadores, es necesario conectorizarlos para su posterior caracterización. De acuerdo con simulaciones realizadas en COMSOL Multiphysics [12], la mejor manera de llevar la señal al resonador es aplicando la tensión directamente sobre el aluminio mediante dos terminales. De esta manera, uno de los terminales es la referencia de tensión mientras que la otra contiene la señal. Por tanto, se han realizado dos agujeros en el aluminio con los que llevar la señal del generador al propio resonador mediante dos cables de cobre esmaltado. Para la sujeción de estos cables, se han estudiado dos aproximaciones diferentes. La primera de ellas es utilizando tornillos roscados con hilo de cobre de 2 o 3 mm de espesor y la segunda es utilizando cilindros metálicos con hilo de cobre de 3 mm.

Para determinar qué opción es la mejor en cuanto a transmisión, se ha realizado una comparación del parámetro S_{21} con cada una de las posibilidades tal y como aparece en la figura 10.

Como se puede observar, la mayor transmisión se consigue con la opción del cilindro metálico, siendo 2 dB superior a las otras dos posibilidades. En la comparación entre el espesor del hilo de 2 o 3 mm con tornillo se demuestra que existe una diferencia entre ambas

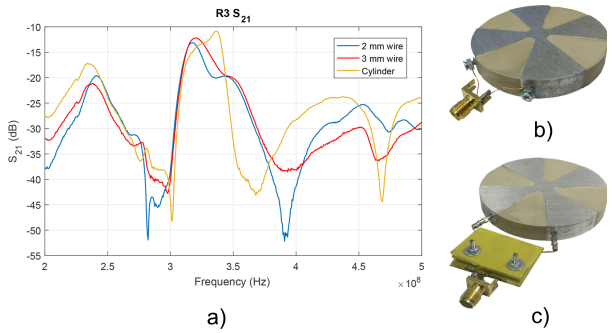

Figura 10: a) Comparación del parámetro S_{21} con tornillo (hilo de 2 mm o 3 mm de espesor) y cilindro. b) Opción con tornillos. c) Opción con cilindros

opciones. Por lo tanto, dependiendo del tipo de conector utilizado, el resonador presenta una respuesta en frecuencia ligeramente diferente. Es decir, el comportamiento del resonador se ve modificado por el modo en el que es conectorizado.

De las tres opciones probadas, la que mejores resultados obtiene es la del cilindro metálico. Por tanto, esta opción ha sido la elegida para la caracterización de todos los resonadores.

Los cables han sido trenzados para reducir las posibles interferencias captadas por los cables entre el conector SMA y el resonador y además, se han apantallado utilizando dos placas paralelas metálicas para disminuir la posible transmisión que hay entre los cables del transmisor y los cables del receptor.

4.3 CARACTERIZACIÓN DE LOS RESONADORES

Para la caracterización de los resonadores se han utilizado dos montajes diferentes. En un primer momento, se ha utilizado un montaje manual con el que es posible desplazar los resonadores de manera horizontal, vertical así como rotar axial y radialmente. Como se ha comentado anteriormente, en la figura 7 se representan los cuatro tipo de medidas diferentes que se han realizado sobre cada resonador.

Más tarde, se ha automatizado la toma de medidas para poder obtener medidas más complejas, como mapas bidimensionales de eficiencia. Por tanto, la caracterización se ha realizado en dos fases. Una fase inicial realizada con montaje manual, correspondiente a la figura 11 izquierda, y más tarde, un sistema de medidas automatizado basado en LabVIEW y etapas motorizadas del fabricante Thorlabs, figura 11 derecha.

En ambos casos, se han realizado medidas de los parámetros S de los resonadores utilizando un analizador de redes vectorial. Las impedancias de los puertos han sido configuradas a 50 Ω.

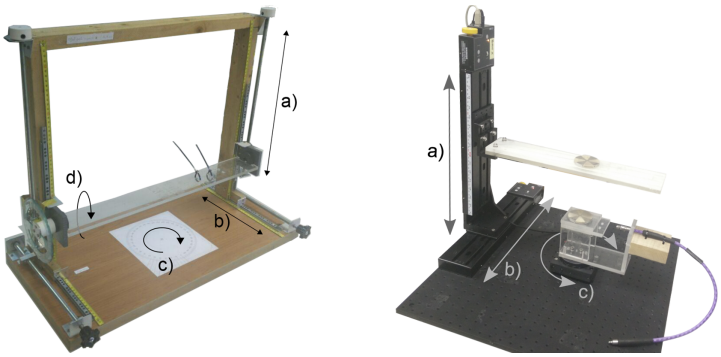

Figura 11: Izquierda: montaje manual. Derecha: montaje automático. a) Desplazamiento vertical. b) Desplazamiento horizontal. c) Rotación radial. d) Rotación axial.

4.3.1 *Automatización de toma de medidas con LabVIEW*

Para la caracterización de los resonadores se ha realizado un montaje con el que automatizar las medidas a tomar. Estas medidas consisten en el mapa bidimensional de la eficiencia en función del desplazamiento vertical y del desplazamiento horizontal, así como del desplazamiento vertical y la rotación radial del resonador, tal y como se esquematiza en la figura 12. De esta manera, se realiza un barrido en cada una de estas dos dimensiones y se calcula la eficiencia de transmisión del resonador en cada punto del espacio. Para ello, se capturan los parámetros S en cada uno de estos puntos a través de un analizador de redes vectorial (ARV).

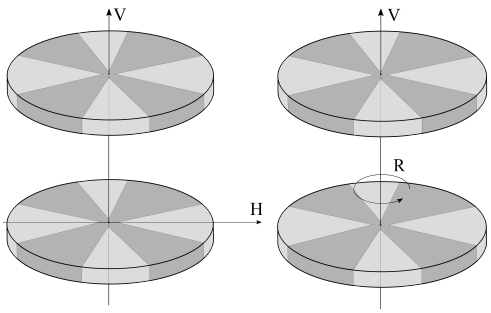

Figura 12: Esquema de medidas de eficiencia. Izquierda: desplazamiento vertical y desplazamiento horizontal. Derecha: desplazamiento vertical y rotación radial.

Los motores que se han utilizado son los modelos LTS300 y NR360S, este último junto al controlador BSC103 de Thorlabs [13]. A través del software LabVIEW de National Instruments [14], se ha creado el programa que mueve los motores, captura los parámetros S utilizando el analizador de redes vectorial ZVA 24 de Rohde&Schwarz [15] y guarda los datos en un fichero de texto. El analizador de redes vectorial está conectado mediante el interfaz GPIB al ordenador y es controlado desde LabVIEW.

Debido a que las medidas a realizar son diferentes, una con movimiento horizontal y otro con rotación, los programas de LabVIEW son ligeramente diferentes. Sin embargo, su funcionamiento a grandes rasgos es muy similar. En la figura 13 se puede observar un esquema general del funcionamiento del programa de LabVIEW.

1. En primer lugar, es necesario inicializar la comunicación con los dos motores y el ARV.

2. Después, el programa entra en dos bucles. El primero mueve uno de los dos motores y el segundo, además de mover el otro motor restante, pide los datos de la medición al ARV y lo añade al final del archivo donde se están guardando las medidas. Esto se repite tantas veces como sea necesario para cubrir todo el espacio requerido.

3. Finalmente, cuando todas las medidas han sido realizadas, los motores vuelven a su posición original y se cierra la comunicación con los motores y el ARV.

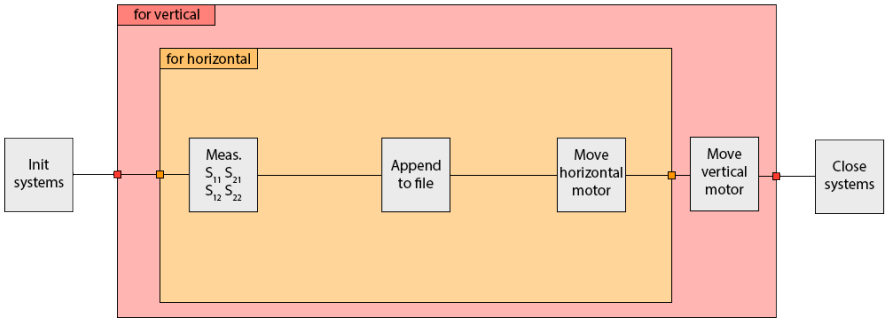

Figura 13: Esquema programa LabVIEW

El formato del archivo donde se guardan las medidas tiene una estructura específica para que su lectura desde MATLAB sea sencilla. El archivo se divide en dos parte, el preámbulo y el cuerpo. En el preámbulo se encuentra información relevante sobre la medida, como por ejemplo el número del resonador medido, el número de puntos barridos o la distancia recorrida. De esta manera, desde MATLAB es posible asignar el tamaño de las variables antes de leer el archivo de manera completa y disminuir su tiempo de procesado. Para determinar dónde termina el preámbulo y dónde empieza el cuerpo, se utiliza la frase *'Start measurement'* como separador. Todas las líneas desde el principio del archivo hasta el separador se consideran parte del preámbulo y todas las líneas desde el separador hasta el final del archivo definen el cuerpo del archivo. La estructura genérica del archivo es la que se muestra en el código 1.

```
Rx - extra info
Param1 sweep: x cm. Param1 sweep points: x
Param2 sweep: x cm. Param2 sweep points: x
Start measurement
Vx Hx
Frequency line
S11 line
S21 line
S12 line
S22 line
Vx Hy
Frequency line
S11 line
S21 line
...
```

Código 1: Estructura del archivo de medidas.

Donde x es un valor entero y Param1 y Param2 pueden ser Vertical, Horizontal o Angular. El cuerpo del archivo está compuesto por una repetición de una estructura más básica que representa una sola medida. Primero se especifican las coordenadas en las que la medida ha sido hecha. Por ejemplo, V3 H-30 representa una distancia vertical de 3 mm y una distancia horizontal de -30 mm respecto al origen. En caso de ser una medida angular, estas claves son Vx Rx, donde la rotación se expresa en grados. A continuación, se añade una fila con las frecuencias a las que se han tomado los puntos de los parámetros S y finalmente se añaden los valores de los parámetros S_{11}, S_{21}, S_{12} y S_{22} en las siguientes filas. Este proceso se repite para cada una de las medidas tomadas. En el código 2 se muestra un extracto de ejemplo de un archivo de medidas.

```
R4 - 1 degree rotation
Horizontal sweep: 20 cm. Horizontal sweep points: 201
Vertical sweep: 15 cm. Vertical sweep points: 151
Start measurement
V5 H-100
2.000E+8 2.015E+8 2.030E+8 [...]
-7.3153E-1 -7.3568E-1 -7.2935E-1 [...]
-3.5196E+1 -3.5279E+1 -3.5160E+1 [...]
-3.5273E+1 -3.5258E+1 -3.5041E+1 [...]
-1.0368E+0 -1.0297E+0 -1.0393E+0 [...]
V5 H-99
2.000E+8 2.015E+8 2.030E+8 [...]
-7.5355E-1 -7.1748E-1 -7.5228E-1 [...]
[...]
```

Código 2: Ejemplo de archivo de medida.

Para poder representar las medidas en un mapa, es necesario procesar el archivo generado con un programa de cálculo numérico como MATLAB. Para ello, el *script* debe leer línea por línea el archivo e ir almacenando los parámetros S en diferentes variables. Para acelerar el procesado del archivo, se utiliza la información extraída del preámbulo para asignar el tamaño de memoria de estas variables. Una vez extraída la información, se crea una matriz bidimensional de los ejes X e Y y se representa el parámetro deseado. En caso de la eficiencia de transmisión entre emisor y receptor, se representará $|S_{21}|^2$ y en caso de la eficiencia de transmisión neta, es decir, sin tener en cuenta la posible desadaptación de los resonadores, se representará $\frac{|S_{21}|^2}{1-|S_{11}|^2}$, tal y como se ha explicado en el apartado 1.1.

4.4 MAPA BIDIMENSIONAL DE EFICIENCIA

Gracias a la automatización de medidas con LabVIEW se ha podido generar unos mapas de eficiencia con los que entender mejor cuál es el comportamiento de los resonadores.

4.4.1 *Desplazamiento horizontal y vertical*

El primer mapa realizado consiste en barrer el espacio realizando un desplazamiento vertical y horizontal del resonador superior. La longitud vertical del desplazamiento ha sido 15 cm, ya que a partir de esta distancia durante la caracterización manual todos los resonadores han mostrado eficiencias muy bajas. En cuanto al desplazamiento horizontal de los resonadores ha sido de 10 cm por cada lado debido a que prácticamente era la máxima excursión posible de los motores

en este eje. En la figura 14 se muestra un esquema de cómo es el barrido total realizado.

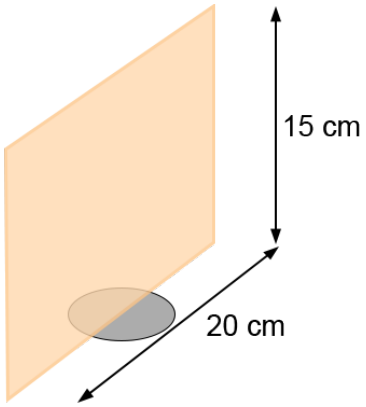

Figura 14: Esquema de barrido vertical y horizontal

Se han obtenido 200 puntos en el eje horizontal y 150 en el eje vertical. Por tanto, la resolución de la medida es de 1 mm en ambos ejes.

De esta forma, el número total de medidas realizadas es de 30.000. Teniendo en cuenta que el proceso de medida tarda aproximadamente unos 2.3 segundos en realizarse, el tiempo total necesario para cubrir todos los puntos es de 19.2 horas aproximadamente.

El resultado de este proceso es el que se aparece en la figura 15, donde se muestra la eficiencia del resonador 5 en función del desplazamiento vertical y horizontal, representado como la eficiencia de transmisión neta.

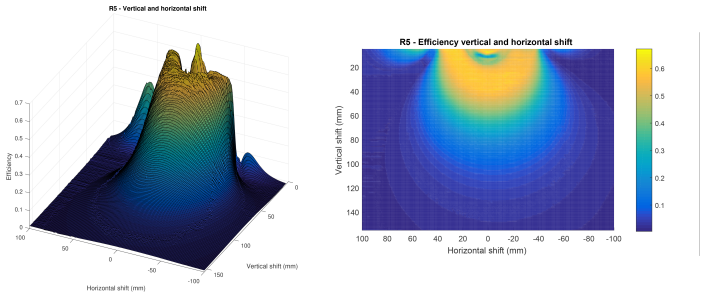

Figura 15: Mapa 3D de la eficiencia del resonador 5 en función del desplazamiento vertical y horizontal

4.4.2 *Desplazamiento vertical y rotación radial*

En el segundo mapa de eficiencia realizado, las variables independientes han sido el desplazamiento vertical y la rotación radial del resonador inferior. La excursión del barrido vertical ha sido el mismo que en el caso anterior, es decir, 15 cm. Para el barrido angular, se han

cubierto los ángulos desde 0 hasta 180º con pasos de 1º, tal y como se muestra en la figura 16. El tiempo total en realizar esta medida completa es de aproximadamente 17.25 horas.

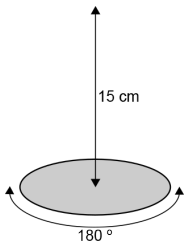

Figura 16: Desplazamiento vertical y rotación radial

En la figura 17 se muestra la eficiencia de transmisión neta del resonador 4 en función de la distancia vertical y el ángulo de rotación. En el anexo A.1 se pueden encontrar el resto de medidas realizadas.

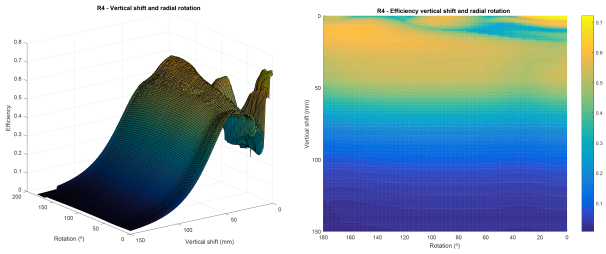

Figura 17: Mapa de la eficiencia del resonador 4 en función del desplazamiento vertical y angular

4.5 MONTAJE DE TRANSMISOR DE LABORATORIO

Debido a que el analizador de redes vectorial solo es capaz de inyectar una potencia baja (1 mW), es necesario recurrir a otro montaje con el que poder realizar pruebas con los resonadores en los que la potencia sea elevada. El montaje propuesto es el que aparece esquematizado en la figura 18.

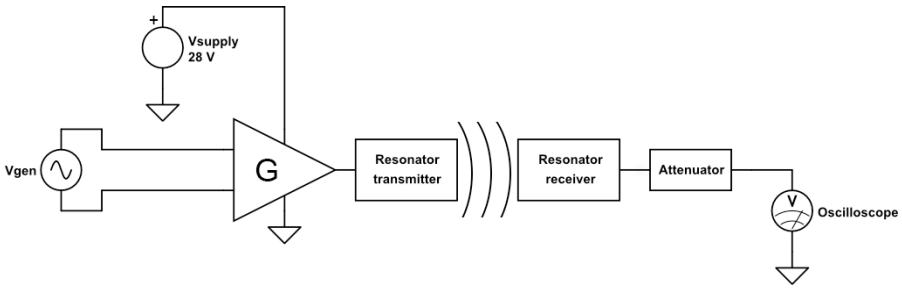

Figura 18: Esquemático del montaje de potencia.

Este sistema consta de un generador de señal de baja potencia con el que obtener una señal sinusoidal a la frecuencia de máxima trans-

ferencia de los resonadores. Esta señal es amplificada mediante un amplificador de potencia de alta frecuencia que deberá estar debidamente alimentado a través de una fuente de alimentación DC. Cuando se ha conseguido la potencia deseada, se inyecta en el resonador transmisor. Una vez captado por el resonador receptor, la señal es llevada a un osciloscopio para poder realizar las medidas. Además, la entrada del osciloscopio está protegida por un atenuador para evitar que potencias elevadas la dañen.

De esta manera, se utilizará un generador de alta frecuencia y un amplificador de potencia con los que generar la señal que se aplicará a los resonadores. El generador de señal utilizado es el Keysight EXG Analog Signal Generator N5171B [16], mientras que el amplificador de potencia es el 1109 - BBM2E5KCL de Empower RF Systems, Inc. [17] capaz de entregar 15 W, cuyo rango de frecuencia de funcionamiento es desde 20 MHz hasta 2500 MHz y con una ganancia aproximada de 40 dB. En la figura 19 se muestra el montaje completo donde se pueden identificar los siguientes elementos:

1. Generador de señal.
2. Fuente de alimentación.
3. Amplificador de potencia.
4. Atenuador de RF.
5. Osciloscopio de amplio ancho de banda.

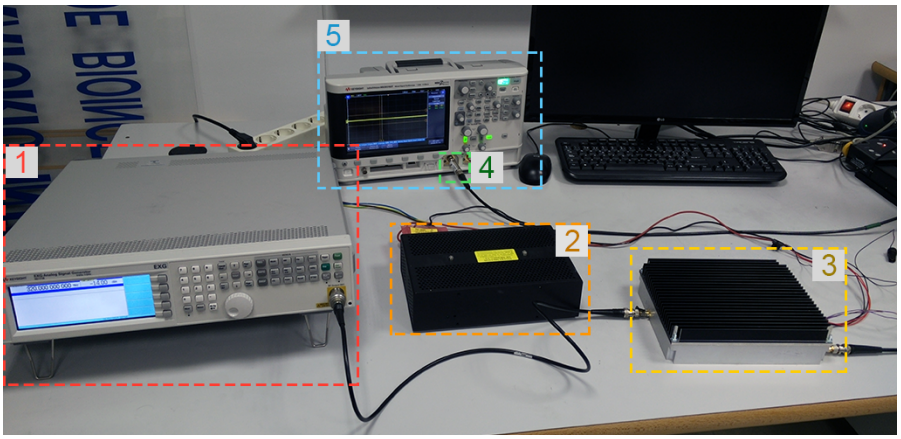

Figura 19: Montaje de transmisor de laboratorio

La impedancia entrada del osciloscopio es de 50 Ω, por lo que se garantiza una buena adaptación entre el receptor y el osciloscopio. La potencia recibida en el osciloscopio a partir de la tensión RMS de la señal corresponde a la expresión 11. Por otra parte, la potencia transmitida se puede calcular como la potencia de la señal generada añadiendo la ganancia del amplificador tal y como se describe en 12.

Por último, con estas dos expresiones es posible calcular la eficiencia de transmisión como la relación entre la potencia recibida y la potencia transmitida 13.

$$P_{received}(dBm) = 10\log_{10}\left(\frac{V_{rms}^2}{Z_{in}}\right) + 30 - L(dB) \tag{11}$$

$$P_{transmitted}(dBm) = P_{gen}(dBm) + G(dB) \tag{12}$$

$$\eta = \frac{P_{received}}{P_{transmitted}} = \frac{10\log_{10}\frac{V_{rms}^2}{Z_{in}} + 30 - L(dB)}{P_{gen}(dBm) + G(dB)} \tag{13}$$

4.6 DISEÑO DEL TRANSMISOR

Con el fin de conseguir un demostrador portátil con el que reemplazar el montaje de laboratorio, se ha decidido crear un transmisor con el que, de la manera más compacta posible, poder inyectar potencia en el resonador transmisor. Debido a que cada uno de los resonadores tiene una frecuencia de resonancia distinta, es necesario elegir un resonador sobre el que realizar el diseño. De lo contrario, realizar un transmisor de banda ancha capaz de excitar con la misma eficiencia implicaría que el ancho de banda fraccional del transmisor es del 24 %. Además, como cada resonador tiene una eficiencia distinta, se ha diseñado el transmisor sobre el resonador con mayor eficiencia.

Se han realizado dos versiones de transmisor diferentes, cada uno tomando como referencia un resonador diferente. En la primera versión se diseño para el resonador R3 y la segundo para el resonador R4.

4.6.1 *Esquema básico*

El transmisor está formado por un oscilador que genera la señal de alta frecuencia y un amplificador que proporciona la potencia necesaria a esta señal. Para adaptar la impedancia de entrada y salida del amplificador se utilizan redes de adaptación, tal y como se muestra en la figura 20.

El circuito se alimentará a través de un conversor de AC/DC de 230V.

4.6.2 *Elección del resonador de referencia*

En el momento de diseñar la primera versión del transmisor solo se contaba con los resonadores R2 y R3. De acuerdo con los datos extraídos de su caracterización, la eficiencia del resonador R3 es superior

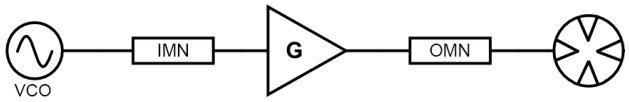

Figura 20: Esquema genérico del transmisor

a la del resonador R2. Por tanto, la primera versión del transmisor se ha realizado sobre el resonador R3.

Del mismo modo, en el momento de diseñar la segunda versión del transmisor, se contaba con un resonador que tenía una mejor eficiencia que el resonador R3. El resonador R4 tiene una eficiencia de 0.5 en un rango de distancias mayor que R3, por lo que se ha escogido como resonador de referencia para la segunda versión del transmisor.

Versión del transmisor	**Resonador**	**Frecuencia (MHz)**
1.0	R3	336
2.0	R4	360

Tabla 2: Resumen de los resonadores de referencia para cada versión de transmisor

4.6.3 *Elección de componentes*

4.6.3.1 *Oscilador controlado por tensión (VCO)*

Para generar la señal a la frecuencia de resonancia, se ha utilizado un oscilador controlado por tensión. El modelo elegido es el JTOS-400+ de Minicircuits [18], cuya frecuencia de oscilación va desde los 200 hasta los 380 MHz. La tensión de alimentación es de 15 V y la corriente que consume es de 20 mA. Por tanto, el consumo del JTOS-400+ es de aproximadamente 0.3 W.

Figura 21: VCO JTOS-400+ de Minicircuits

Para caracterizar el componente, se ha aplicado una tensión de alimentación de 15 V y se ha realizado un barrido de la tensión de entrada desde 0 V hasta 17 V con pasos de 0.5 V. Posteriormente, se ha calculado la potencia del armónico fundamental para obtener la curva de potencia para cada una de las frecuencias, así como la distorsión armónica total calculada según la expresión 14, donde v_{a_i} es la amplitud de la tensión del i-ésimo armónico y v_f es la amplitud

de la tensión de armónico fundamental. Como se puede observar en la figura 22, la variación de frecuencia de salida en función de la tensión de entrada es lineal. Sin embargo, la potencia de salida varía en función de la frecuencia de salida, estando entre los 17.14 dBm y los 7.77 dBm. Por tanto, hay una variación de 9.37 dB entre la máxima potencia de salida y la mínima potencia de salida del oscilador. Para conseguir la frecuencia de salida de 336 MHz, la tensión de entrada del oscilador debe ser de 11.55 V, y para 360 MHz es de 12.8 V.

$$\mathrm{THD(dBc)} = 10\log_{10}\left(\frac{\sum_i v_{a_i}^2}{v_f^2}\right) \tag{14}$$

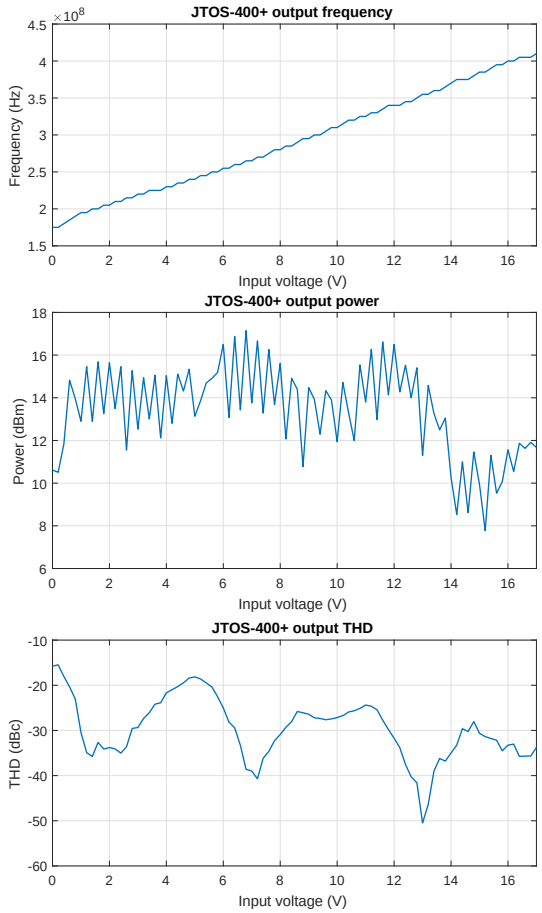

Figura 22: Arriba: frecuencia de salida en función de la tensión de entrada. Medio: potencia de salida en función de la tensión de entrada. Abajo: THD en función de la tensión de entrada

4.6.3.2 *Amplificador RF*

Para la etapa de amplificación, se ha elegido el integrado TQP7M9105 de Qorvo [19], cuya tensión de alimentación es de 5 V, una corriente de polarización de 245 mA y una ganancia aproximada de 14 dB

para un dispositivo sin adaptar en la banda de frecuencia requerida. Este integrado tiene un encapsulado SOT-89, tal y como se muestra en la figura 23. Teniendo en cuenta la tensión de alimentación y la corriente máxima de polarización, la potencia máxima consumida es de 1.225 W.

Figura 23: Amplificador TQP7M9105

4.6.3.3 *Adaptador de AC/DC*

Para alimentar el circuito se utilizará un adaptador AC/DC que convierta los 230 V_{rms} de la red eléctrica en una tensión continua V_{DC}. Dado que la tensión de alimentación del oscilador controlado por tensión (VCO) es de 15 V, se ha elegido esta tensión como la tensión de salida del adaptador de AC. Por tanto, $V_{DC} = 15$ V. El adaptador AC/DC elegido es el PSAC12R-150 de Phihong [20], cuya tensión de rizado es de 150 mV.

La potencia que puede entregar el adaptador es 12 W. Esto se traduce en que la corriente que es capaz de proporcionar es 800 mA. Dado que los componentes que más consumo requieren son el amplificador y el oscilador, cuyo consumo combinado es de aproximadamente 1.525 W y ninguno exige una corriente superior a la de 800 mA, el adaptador cumple los requisitos de potencia que el circuito demanda.

4.6.3.4 *Regulador lineal*

La alimentación del amplificador TQP7M9105 es de 5 V. Por tanto, es necesario reducir la tensión de salida del adaptador AC/D. Para ello, se ha utilizado un regulador lineal L78M05 junto a un disipador para evacuar el calor que se produce al regular la tensión. De acuerdo con [21], la eficiencia de un regulador lineal depende de la tensión de entrada y de salida. Por tanto, según con la expresión 15, la eficiencia del regulador será del 33 %.

$$\eta = \frac{P_{output}}{P_{output} + P_{loss}} = \frac{V_o \cdot I_o}{V_o \cdot I_o + (V_{in} - V_o) \cdot I_o} = \frac{V_o}{V_{in}} = \frac{5}{15} = 0{,}33 \tag{15}$$

4.6.3.5 *Convertidor DC/DC conmutado*

Dado que en la primera versión del transmisor se ha utilizado un regulador lineal cuya eficiencia era suficientemente baja (33 %) como para suponer un problema de disipación térmica, en la segunda versión del transmisor se ha utilizado un convertidor DC/DC conmutado y aislado tipo Flyback para regular la tensión V_{DC} a 5 V y así alimentar el amplificador. Dado que la eficiencia de conversión de este tipo de circuitos es de >80 %, se soluciona el problema de disipación térmica.

El modelo elegido es el EC3SAW-24S05P de Cincon [22], capaz de entregar y 5 V a la salida y 600 mA de corriente.

4.6.3.6 *Conclusión*

Teniendo en cuenta los componentes elegidos, el esquema del transmisor es el que se muestra en la figura 24.

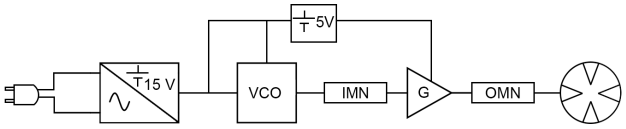

Figura 24: Esquema final del transmisor

En la tabla 3 se resumen los componentes utilizados en el transmisor, en el que se diferencia el regulador utilizado en la versión R3 del transmisor del de la versión R4.

Etapa	Componente
Adaptador AC/DC	PSAC12R-150
VCO	JTOS-400+
Amplificador	TQP7M9105
Regulador v1.0	L78M05
Regulador v2.0	EC3SAW-24S05P

Tabla 3: Resumen de componentes del transmisor

4.6.4 *Stack-up*

Para poder realizar el diseño del transmisor, primero es necesario definir el *stack-up* sobre el que se va a implementar la placa de circuito impreso (PCB). Esto es debido a que las características eléctricas de las pistas varían en función del dieléctrico utilizado o el número de capas. Para ello, se ha utilizado un *stack-up* de dos capas con un dieléctrico FR4 IS-400 de Isola [23], cuyo grosor es de 1.55 mm, su permitividad dieléctrica a 500 MHz es de $\epsilon = 3{,}9$ y su tangente de

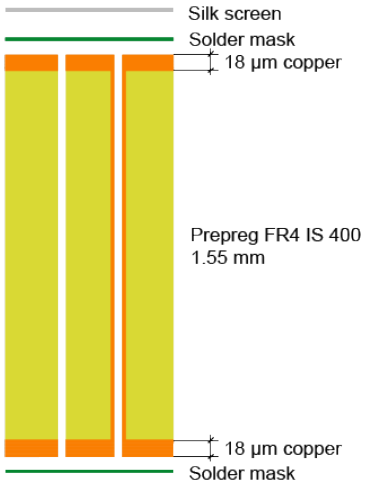

Figura 25: *Stack-up* de la placa de circuito impreso (PCB) de los transmisores

pérdidas $\tan\delta = 0{,}022$. Las capas de cobre tienen un espesor de 18 µm, por lo que el grosor total es de 1.568 mm. Las capas de cobre están recubiertas por una capa de *solder mask* verde y finalmente la capa superior está serigrafiada.

Con esta configuración el ancho de pista necesario para que la impedancia característica sea 50Ω es de $W_{50} = 3{,}1599\text{mm}$ de acuerdo con TXLine de AWR Design Environment.

4.6.5 *Versión R3*

4.6.5.1 *Diseño con AWR Environment*

El comportamiento frecuencial del resonador varía con las condiciones operativas de los resonadores, es decir, cambia con la distancia a la que se encuentran el transmisor y el receptor debido a la alteración del acoplamiento magnético entre ellos. En la figura 26 se muestra la variación de la frecuencia de máxima transferencia de energía en función de la distancia vertical y horizontal entre ambos resonadores. La diferencia entre la frecuencia más alta y más baja a la que se da la máxima transferencia de energía es de 206 MHz. Sin embargo, en la zona donde se puede realizar una transferencia de energía efectiva debido a que la eficiencia no es baja, la frecuencia está alrededor de 336 MHz (zona central de la figura 26). Es por ello que 336 MHz ha sido la frecuencia central con la que se ha diseñado el transmisor.

Tal y como se ha comentado y se muestra en la figura 20 y 24, entre el VCO y el amplificador así como entre el amplificador y el resonador, existe una red de adaptación de impedancias. Analizando los parámetros S del amplificador que proporciona el fabricante se puede observar la desadaptación de impedancia tanto en la entrada como en la salida. En la figura 27 se representan los parámetros S_{11} y S_{22} entre 200 MHz y 500 MHz, donde la progresión ascendente de

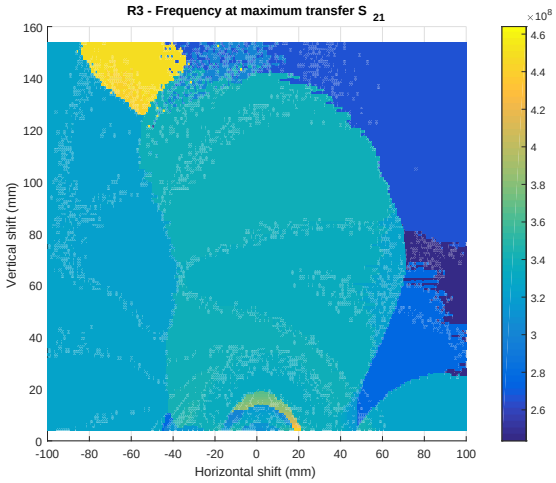

Figura 26: Distribución de las frecuencias de máxima transferencia en función de la distancia vertical y horizontal

la frecuencia sigue el sentido dextrógiro. Por tanto, debido a que la impedancia de entrada y salida del amplificador TQP7M9015 no está adaptada a 50 Ω, es necesario realizar una adaptación para asegurar la máxima transferencia de potencia.

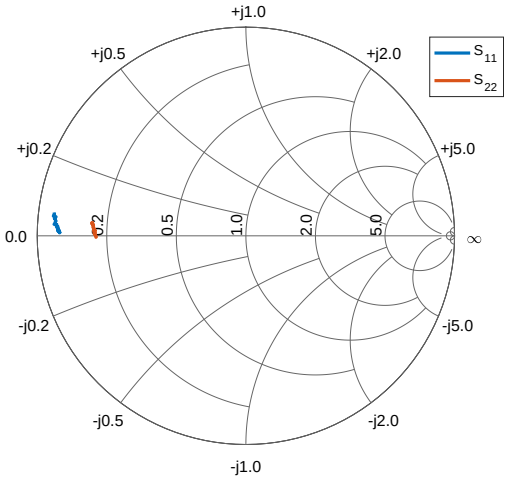

Figura 27: Parámetros S_{11} y S_{22} del amplificador TQP7M9105 entre 200 MHz y 500 MHz.

La red de adaptación entre el VCO y el amplificador tiene una estructura en escalera de dos etapas con los condensadores en serie y las inductancias en derivación, conformando un filtro paso alto, tal y como se observa en la figura 28. En ella se muestra la red de adaptación de entrada implementada con líneas microstrip y componentes reales (condensadores e inductores) de Murata. Para simular los componentes se han utilizado los parámetros S proporcionados por el fabricante. También se ha utilizado la anchura de las pistas como

variables extra en la optimizacion de la red para disponer de más grados de libertad.

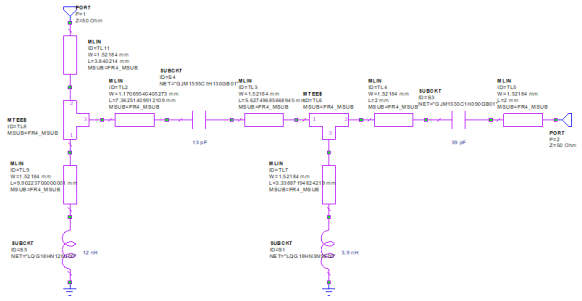

Figura 28: Red de adaptación de entrada real

En la 29 se observa cómo las pérdidas de retorno a la entrada están por debajo de -15 dB entre las frecuencias de 260 y 350 MHz.

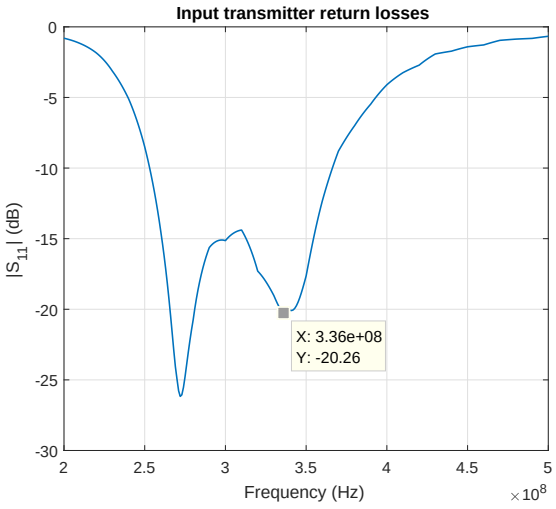

Figura 29: Pérdidas de retorno a la entrada del transmisor R3. Ver tabla 1

En cuanto a la red de adaptación de salida, se ha utilizado una red en L con el condensador en serie y el inductor en derivación. De esta manera, se filtra la componente DC ya que el amplificador se alimenta a través del pin de salida. La metodología de diseño utilizada es la misma que para la red de adaptación de la entrada.

En la figura 31 se muestran las pérdidas de retorno a la salida del transmisor R3. Estas son menores de -15 dB entre 274 y 348 MHz.

Con esta configuración, el parámetro S_{21} del transmisor es el que se muestra en la figura 32. A la frecuencia central de trabajo (336 MHz), la ganancia del amplificador es de 21 dB. Sin embargo, debido a que la potencia de salida está próxima al punto de compresión a 1 dB (mínimo 28.7 dBm) se espera que la ganancia sea menor ya que el programa de diseño no tiene en cuenta estas características.

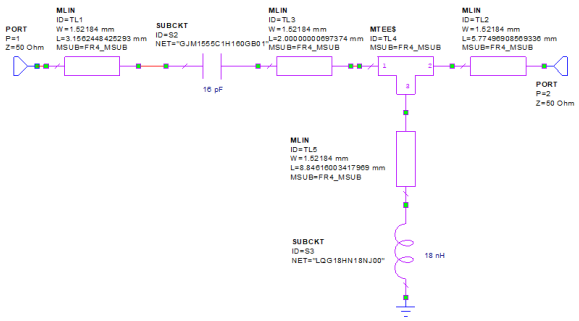

Figura 30: Red de adaptación de salida real

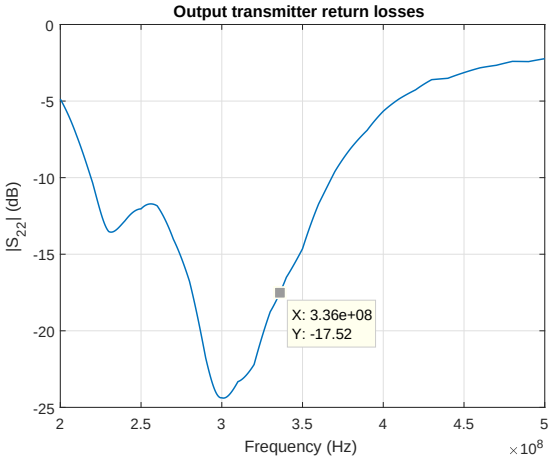

Figura 31: Pérdidas de retorno a la salida del transmisor R3. Ver tabla 1

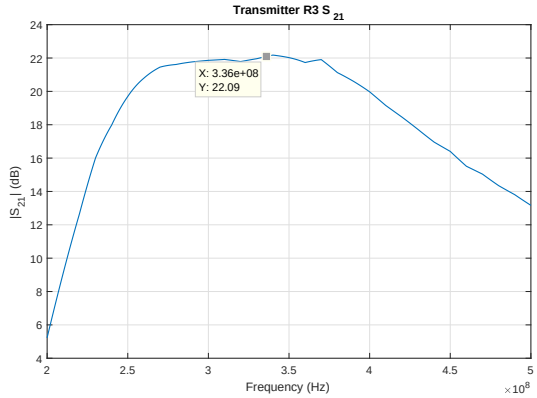

Figura 32: Parámetro S_{21} del transmisor R3 con las redes de adaptación reales. Ver tabla 1

En cuanto a la red de alimentación, se han utilizado condensadores de desacoplo que eliminen fluctuaciones de la tensión de alimentación de baja frecuencia en la alimentación del amplificador. Estos condensadores se colocan lo más cerca posible de la alimentación. De esta manera, se cargan a la tensión de alimentación y son ca-

paces de entregar carga al amplificador. Con esto se consigue que se independice el diseño de la inductancia de las pista y los cables, proporcionando una fuente de carga de baja impedancia entre V_{cc} y masa. Esta inductancia de las pistas y los cables se opone a cambios bruscos de corriente, por lo que sin condensadores de desacoplo, las demandas instantáneas de corriente producirían caídas importantes en la tensión de alimentación. Aunque estas demandas instantáneas de corriente son más típicas de circuitos digitales, se realizará el diseño siguiendo la hipótesis de que se puedan dar para otorgar más robustez al diseño.

Para calcular el valor aproximado de los condensadores, es necesario conocer la máxima variación de tensión que soporta el amplificador y la variación de corriente. De acuerdo con el datasheet del amplificador TQP7M9105, la máxima variación de tensión permitida es de 0.25 V por encima de los 5 V. Por tanto, se escogerá como un valor de referencia $\Delta V = 0{,}25$ V. En cuanto a la corriente, se supondrá que la variación máxima será la de la corriente de polarización máxima, que de acuerdo con el datasheet es de 220 mA. Por tanto, $\Delta I = 220$ mA. De esta manera, es posible calcular la reactancia máxima permitida entre masa y alimentación de acuerdo con la expresión 16.

$$X_{max} = \frac{\Delta V}{\Delta I} = \frac{0{,}25\ \text{V}}{220\ \text{mA}} = 1{,}13\ \Omega \tag{16}$$

En cuanto a la inductancia de la pista y los cables, se ha supuesto el valor de la inductancia de *choke* que es de 330 nH con un factor de seguridad de 2. Por tanto, L = 660 nH. De esta manera, la frecuencia a la cual esta inductancia presenta esta reactancia es 274 kHz, de acuerdo con la expresión 17.

$$f_{ind} = \frac{X_{max}}{2\pi \cdot L} = \frac{1{,}13}{2\pi \cdot 660 \cdot 10^{-9}} \approx 274\ \text{kHz} \tag{17}$$

De esta manera, el valor nominal de un condensador que tenga la reactancia X_{max} a la frecuencia f_{ind} es de 255 nF de acuerdo con la expresión 18.

$$C_{bulk} = \frac{1}{2\pi \cdot f_{ind} \cdot X_{max}} = \frac{1}{2\pi \cdot 548 \cdot 10^3 \cdot 1{,}13} = 511\text{nF} \tag{18}$$

Por tanto, la capacidad de desacoplo mínima debe ser de al menos 0.5 µF.

La red de desacoplo de 5 V es la que se muestra en la figura 33 donde hay tres condensadores en paralelo de valor 1 µF, 100 nF y 2,2 pF y en la que el inductor de *choke* es de 330 nH.

Para calcular la efectividad de la red de adaptación se ha estudiado la impedancia de entrada de la red a diferentes frecuencias utilizando los parámetros S de los componentes proporcionados por el

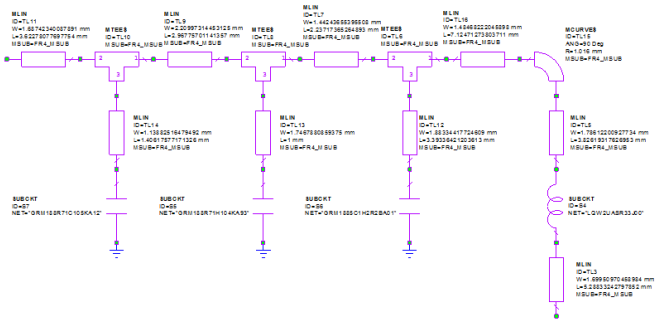

Figura 33: Red de desacoplo real de la alimentanción de 5 V del amplificador

fabricante. El resultado es el que se observa en la figura 34 donde la impedancia de entrada se mantiene por debajo de 1 Ω hasta la frecuencia de 1.1 GHz aproximadamente y por debajo de 2 Ω hasta 2 GHz. Por tanto, se considera un camino de baja impedancia para altas frecuencias aceptable. La red tiene su mínimo de impedancia en 9 MHz con un valor de 0.01 Ω debido a la frecuencia de autoresonancia entre la capacidad equivalente y las inductancias parásitas de los componentes. A partir de esta frecuencia predomina el efecto parásito de la inductancia ya que, como se puede observar, la impedancia equivalente aumenta con la frecuencia.

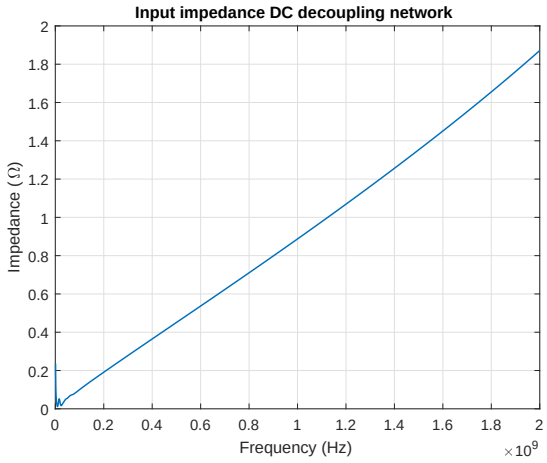

Figura 34: Impedancia de entrada de la red de desacoplo del amplificador

El resto del esquemático, consistente en la interconexión del VCO y los conectores de alimentación, puede encontrarse en el anexo.

En la figura 35 se muestra una representación de la PCB del transmisor R3, en el que los componentes SMD tienen un tamaño 0402, 0603 y 1008.

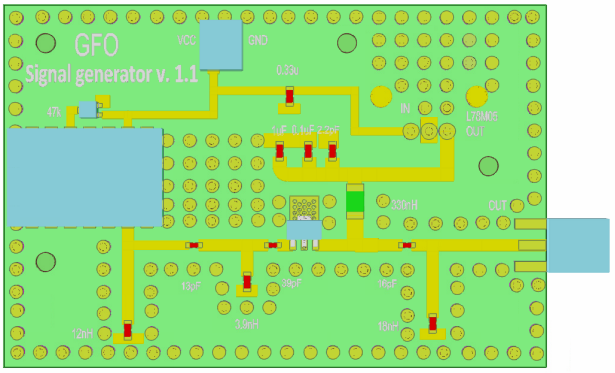

Figura 35: Visualización de la PCB del transmisor R3

4.6.5.2 *Resultado*

La placa de circuito impreso (PCB) ha sido fabricado por Eurocircuits utilizando el *stack-up* referido anteriormente. El resultado es el que aparece en la figura 36 donde se muestra el anverso y el reverso de la PCB.

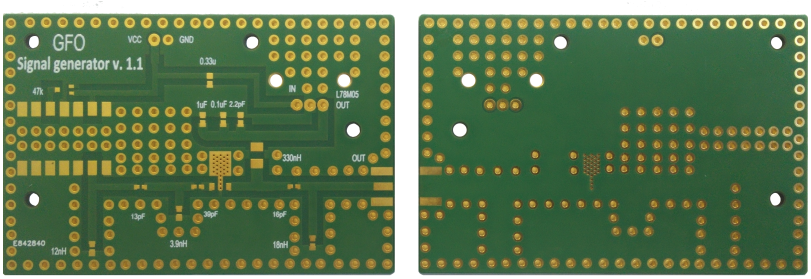

Figura 36: Anverso y reverso de la PCB del transmisor R3

Para caracterizar el transmisor R3 se ha realizado un barrido en frecuencia variando la tensión aplicada al oscilador controlado por tensión. La señal de salida se ha capturado utilizando un osciloscopio Keysight Infiniivision MSOX3102T [24] con capacidad de capturar señales de hasta 1 GHz. Con la señal capturada en el dominio temporal, se ha calculado la FFT para poder cuantificar la potencia del armónico fundamental. De esta manera, es posible obtener la potencia entregada por el transmisor para distintas frecuencias teniendo en cuenta los efectos no lineales del amplificador. El osciloscopio se ha configurado para que guarde 25000 muestras por archivo, con una frecuencia de muestreo de 5 GSa/s. Por tanto, la resolución frecuencial que se puede obtener es de 200 kHz de acuerdo con la expresión 19.

$$\Delta f = \frac{F_s}{N_{samples}} = \frac{5 \cdot 10^9 \text{ Hz}}{25000} = 200 \text{ kHz} \tag{19}$$

El resultado obtenido es el que se muestra en la figura 37, en el que se ve que para 336 MHz, la potencia total de salida es de 0.56 W, mientras que la potencia del armónico fundamental es de 0.5 W.

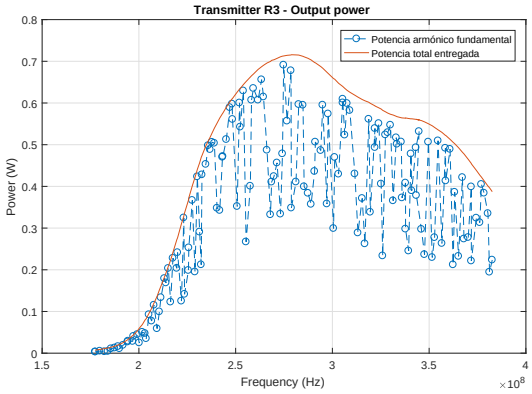

Figura 37: Potencia de salida total y de armónico fundamental del transmisor R3

También se puede observar como la potencia del armónico fundamental varía mucho en función de la frecuencia de salida, mientras que la potencia total entregada tiene una evolución menos abrupta. Esto es debido a la variación de potencia de salida del oscilador JTOS-400+ y sobretodo a las no linealidades del amplificador TQP7M9105.

Por último, se han tomado imágenes térmicas del transmisor para poder comparar el rendimiento térmico entre el transmisor R3 y el transmisor R4. La cámara térmica es el modelo T de Flir. En este caso, no es posible conocer la temperatura absoluta de la superficie del transmisor debido a la imposibilidad de encontrar datos fiables acerca de la emisividad del *solder mask* que recubre toda la PCB. Sin embargo, dado que los materiales plásticos tienen una alta emisividad se ha escogido un valor de 0.9 con el que tomar las imágenes térmicas. Con esta configuración, las imágenes conseguidas son las que aparecen en la figura 38, donde se muestra la parte frontal y trasera del transmisor.

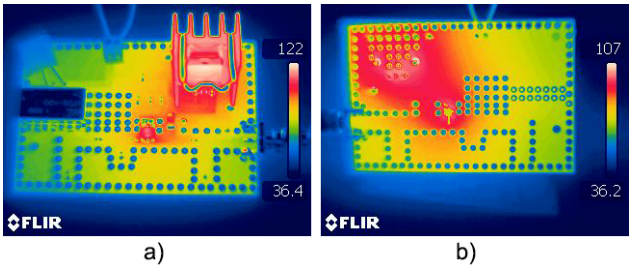

Figura 38: a) Parte frontal del transmisor R3. b) Parte trasera del transmisor R3

Como era previsible, los componentes más calientes son el regulador lineal y el amplificador. Los mecanismos de evacuación de calor en el circuito en este caso tres. En primer lugar por conducción. Este

puede apreciarse si se presta atención a la parte trasera de la PCB, donde se ve cómo las vías térmicas transmiten el calor al cobre de la parte trasera. El otro mecanismo existente es la convección, ya que el calor del disipador del regulador lineal es transferido al aire de su alrededor, así como al aire alrededor de la PCB. Para finalizar, el último mecanismo de evacuación de calor es el de radiación, dado que los componentes están a una temperatura superior que el resto del entorno.

4.6.6 *Versión R4*

El procedimiento seguido para el diseño de la segunda versión del transmisor es la misma que la de la primera. Sin embargo, al cambiar de resonador los parámetros eléctricos como son la constante dieléctrica de la cerámica o la frecuencia de resonancia varían. Por tanto, es necesario rediseñar las redes de adaptación de entrada y de salida para garantizar una buena transmisión de potencia entre el oscilador y el amplificador y entre el amplificador y el resonador. La frecuencia de resonancia a la que se da una máxima transferencia de energía en este caso es de 360 MHz.

Además, tal y como se ha mencionado en el apartado 4.6.3.5, en esta versión del transmisor se ha utilizado un convertidor DC/DC conmutado debido a la baja eficiencia del regulador lineal y los problemas térmicos que se derivaban.

4.6.6.1 *Diseño con AWR Environment*

Del mismo modo que en la versión del transmisor R3, el transmisor R4 cuenta tanto de una red de adaptación a la entrada y a la salida por lo motivos comentados anteriormente. En este caso, la red de adaptación de salida está formada por una red en L paso alto con un inductor en serie y un condensador en derivación. Las anchuras de las pistas están fijadas de manera que su impedancia característica sea de 50 Ω. Por tanto, con las características del *stack-up* descritas en el apartado anterior, su anchura es de 3.188 mm.

La implementación real de la red de adaptación de entrada es la que se muestra en la figura 39.

Del mismo modo, la red de adaptación de salida se muestra en la figura 40.

Con estas redes de adaptación, el circuito presenta las pérdidas de entrada y salida de la figura 41. Se puede observar como las pérdidas de retorno de entrada tienen un ancho de banda de 67 MHz con unas pérdidas de retorno menores de -10 dB y 25.7 MHz por debajo de -15 dB. En cuanto a las pérdidas de retorno de salida, el ancho de banda a -10 dB es de 140.9 MHz mientras que a -15 dB es de 54.9 MHz. El mínimo de las pérdidas de retorno tanto de entrada como de salida está centrado en la frecuencia de resonancia de 360 MHz.

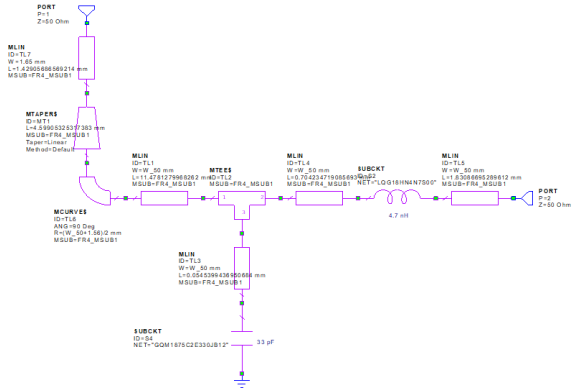

Figura 39: Red de adaptación real de entrada del transmisor R4. Ver tabla 1

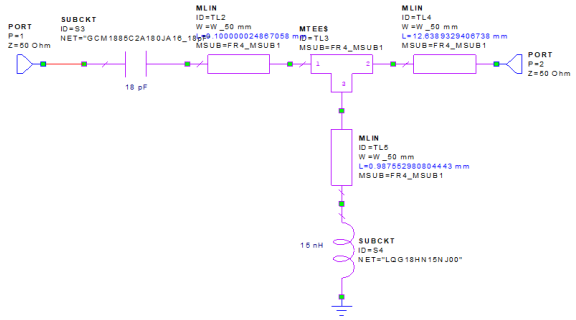

Figura 40: Red de adaptación real de salida del transmisor R4. Ver tabla 1

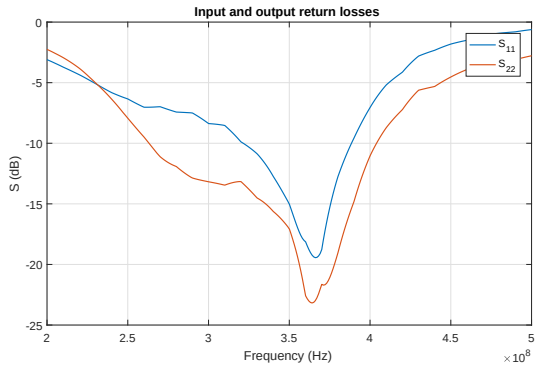

Figura 41: Pérdidas de retorno de entrada y salida del transmisor R4. Ver tabla 1

Por último, en la figura 42 aparece el parámetro S_{21} del transmisor R4. En este caso, todos los componentes tienen un encapsulado 0603 excepto la inductancia de *choke* para alimentar el amplificador cuyo encapsulado es 1008.

Las redes de desacoplo de continua son idénticas a las utilizadas en el transmisor R3 con la excepción de que se ha añadido un condensador electrolítico de 200 µF para filtrar todavía más las variaciones de

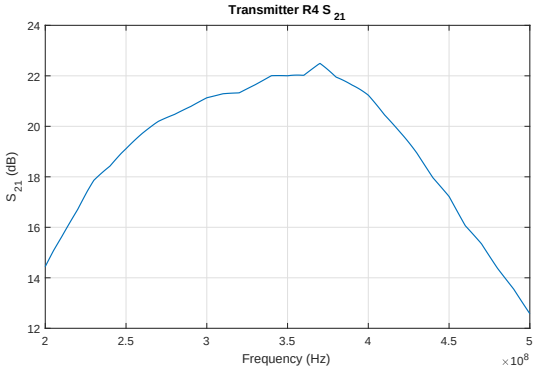

Figura 42: Parámetro S_{21} del transmisor R4. Ver tabla 1

baja frecuencia de la tensión de alimentación. Además, también se ha añadido otra red de desacoplo de continua en la entrada de 15 V para asegurar que el rizado de la tensión de alimentación del oscilador sea bajo. Por último, también se ha añadido un LED SMD para indicar cuándo el transmisor está alimentado.

En la figura 43 se representa el modelo 3D de la PCB completa.

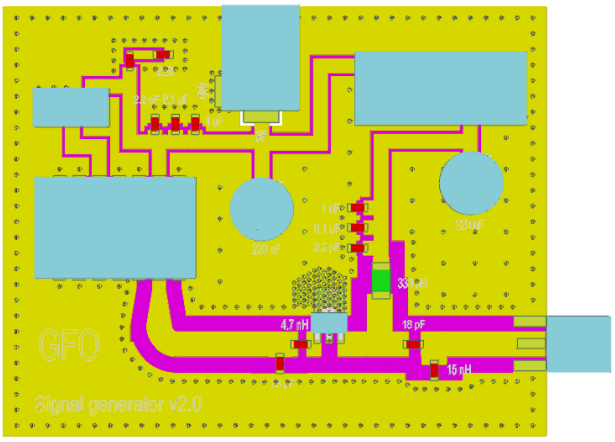

Figura 43: Visualización de la PCB del transmisor R4

4.6.6.2 *Resultado*

Analizando del mismo modo que el transmisor R3 la señal de salida, se obtiene que la potencia del armónico fundamental a la salida es de 0.655 W. Las no linealidades del amplificador se vuelven a mostrar, en este caso haciendo que la potencia del armónico fundamental varíe de forma notable con incrementos pequeños de frecuencia.

Comparando la imagen térmica de la figura 45 con la figura 38, se puede observar como el problema térmico del regulador lineal ha desaparecido. Para poder compararlas, se ha configurado la cámara con la misma emisividad que la utilizada en la imagen anterior. El convertidor DC/DC conmutado es más eficiente y su disipación de calor

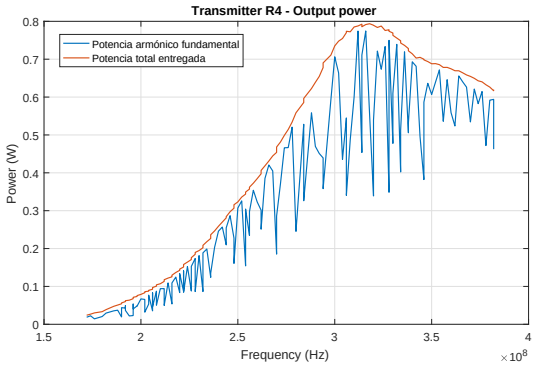

Figura 44: Potencia de salida total y del armónico fundamental del transmisor R4

ya no es importante. En este caso el amplificador es el componente que más potencia disipa y como se puede observar, la conducción de calor a la parte trasera de la PCB es efectiva gracias a las vías térmicas realizadas en el *pad* del amplificador.

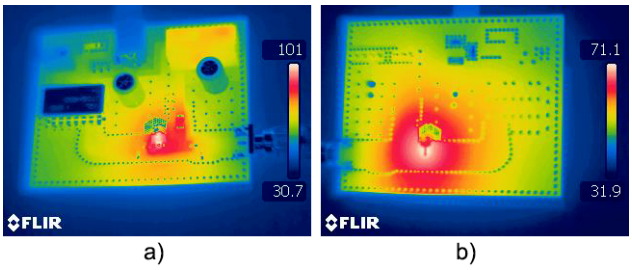

Figura 45: a) Parte frontal del transmisor R4. b) Parte trasera del transmisor R4

Parte V

PLIEGO DE CONDICIONES

5 PLIEGO DE CONDICIONES

5.1 PRESUPUESTO

El coste del proyecto se puede desglosar en tres partes: la amortización de equipos, los gastos de material fungible y los costes de mano de obra. Para elaborar este presupuesto se han seguido las recomendaciones del CTT (Centro de apoyo a la innovación, la investigación y la transferencia de tecnología) de la Universitat Politècnia de València.

5.1.1 *Amortización de equipos*

El coste de amortización se ha calculado de acuerdo con la Ec. 20.

$$\text{Coste amortización} = \text{precio} \cdot \frac{\text{tiempo de uso}}{\text{período de amort.}} \cdot \%\ \text{de uso} \quad (20)$$

Los equipos utilizados durante el proyecto han sido:

- Analizador de redes vectorial ZA-24 Rohde&Schwarz
- Osciloscopio Keysight MSOX3102T
- Generador de señal Keysight N5171B EXG X-Series
- Ordenador de sobremesa
- Amplificador RF Empower 1109
- Motores Thorlabs:
  - 2 motores LTS300
  - Z-Axis Bracket
  - 1 motor angular NR360
  - 1 controlador BSC103

Concepto	Precio (€)	Tiempo de uso (meses)	Periodo de amortazación (meses)	Porcentaje de uso	Coste (€)
NVA	60000	10	120	0,8	4000
Osciloscopio	9926,53	10	60	0,4	661,77
Ordenador	3000	10	60	1	500
Amplificador RF	7961,8	10	60	0,1	132,7
Generador de señal	6790,2	10	60	0,1	113,17
Motores Thorlabs	12819	10	60	0,4	854,6
				Total	6262,24 €

Tabla 4: Tabla resumen de amortización de equipos

Por tanto, los gastos de amortización del proyecto han sido de 6262.24 €.

5.1.2 *Gastos de material fungible*

De acuerdo con las recomendaciones del CTT, se considera como material fungible los prototipos realizados durante el proyecto. Por tanto, se incluyen en este apartado los materiales necesarios para la implementación de los transmisores. Entre ellos se encuentran los componentes electrónicos y las PCB encargadas de fabricar en la empresa Eurocircuits. A continuación se detallan los pedidos a proveedores y empresas que se han realizado durante el transcurso del proyecto acerca de componentes electrónicos. Los principales distribuidores son Mouser, RS Amidata y Minicircuits. Además, también se contemplan de manera aproximada otros gastos como papel y tinta de impresora utilizada durante el transcurso del proyecto por completitud del presupuesto. En el apartado A.2 del anexo se incluye el desglose de los pedidos realizados.

Pedido	**Fecha**	**Cantidad (€)**
Minicircuits	20/07/2016	409,58
RS Amidata	01/07/2016	33,51
Eurocircuits	05/09/2016	68,08
Mouser	22/07/2016	50,74
Mouser	26/08/2016	54,18
Mouser	21/03/2017	89,30
Eurocircuits	30/03/2017	69,93
RS Amidata	23/06/2017	39,01
Papel		3
Tinta		30
	Total	847,33 €

Tabla 5: Tabla resumen de gastos de material fungible

5.1.3 *Gastos de mano de obra*

Los gastos de mano de obra del proyecto se dividen en dos etapas, ya que la contratación en el Grupo de Fenómenos Ondulatorios ha sido en un primer momento desde finales de mayo de 2016 hasta agosto de 2016 en la que se realizaban 2 horas diarias y posteriormente desde enero de 2017 hasta julio de 2017 en las que se ha trabajado 5 horas diarias. De acuerdo con las cantidades orientativas proporcionadas por el CTT sobre los costes indicativos de personal de plantilla en proyectos, el coste horario de un técnico de laboratorio es de 16.58 €/h.

Coste horario (€/h)	Horas diarias	Meses	Coste (€)
16,58	2	3	2984,4
16,58	5	7	17409
		Total	20393,4 €

Tabla 6: Tabla resumen de gastos mano de obra

Por tanto, los gastos de mano de obra durante los 10 meses del proyecto han sido de 20393.4 €.

5.1.4 *Conclusiones*

Teniendo en cuenta la amortización de equipos, los gastos de material fungible y los gastos de mano de obra, el presupuesto del proyecto ha sido de 27502.97 €. En la tabla 7 se resume el total de gastos.

Concepto	Coste (€)
Amortización de equipos	6262.24
Material fungible	847,33
Mano de obra	20393,4
Total	27502.97

Tabla 7: Tabla resumen de gastos totales del proyecto

Parte VI

CONCLUSIONES Y FUTURO DESARROLLO

6 CONCLUSIONES Y FUTURO DESARROLLO

6.1 CONCLUSIONES

Durante la realización de este proyecto se han cumplido todos los objetivos iniciales marcados. Los resonadores han sido montados y caracterizados con éxito a través del montaje manual y posteriormente el automatizado mediante etapas motorizadas. Además, se han podido extraer correctamente las gráficas necesarias para la elaboración del artículo científico relacionado con la investigación y han ayudado ampliamente a entender el comportamiento de este tipo de estructuras. Una de las conclusiones que se extraen es la alta dependencia que existe entre la tangente de pérdidas de la cerámica y la eficiencia. En el caso del resonador R6, para una tangente de pérdidas de 0.6 %, no se ha conseguido una trasferencia de potencia efectiva a ninguna distancia ni frecuencia, cuya eficiencia en ningún caso superaba el 8 %. Por tanto, la cerámica debe tener una tangente de pérdidas menor para que la transferencia de energía sea efectiva. Para el resto de resonadores, se ha podido observar que en todos ellos la respuesta de la eficiencia tiene un comportamiento similar con una eficiencia baja cuando ambos resonadores están muy próximos y un máximo de transferencia a una distancia alrededor de los 4 cm para todos ellos.

En cuanto al diseño de la electrónica de transmisión, también se han conseguido buenos resultados, haciendo posible la construcción del demostrador tecnológico portátil que se muestra en la figura 46. Sin embargo, queda espacio para su mejora y es uno de los puntos propuestos en el futuro desarrollo del proyecto.

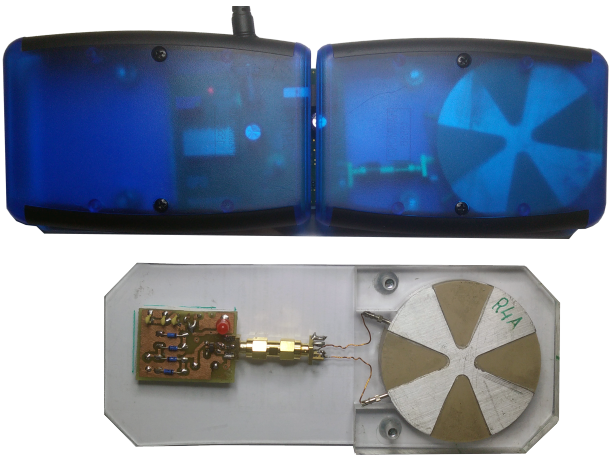

Figura 46: Demostrador tecnológico de transferencia inalámbrica de potencia mediante resonadores metalodieléctricos

Con todo, la potencia que se ha podido inyectar en el demostrador tecnológico al resonador es de 0.655 W. En la figura 47 se muestra la potencia recibida en el demostrador tecnológico medida con el osciloscopio cuyo puerto está configurado a una impedancia de entrada de 50 Ω. La máxima potencia recibida es de 0.2768 W a una distancia de aproximadamente 4 cm. Más allá de este punto, la potencia recibida disminuye de acuerdo con el comportamiento estudiado. En la tabla 8 se detallan los valores de potencia recibida.

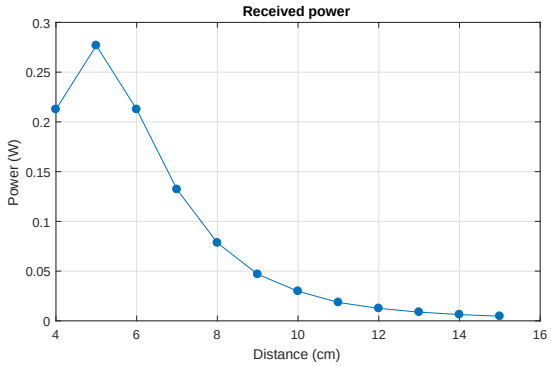

Figura 47: Potencia recibida con el demostrador tecnológico a diferentes distancias

Distancia (mm)	Potencia (mW)
40	212,56
50	276,77
60	212,562
70	132,1
80	78,41
90	46,82
100	29,77
110	18,47
120	12,55
130	8,76
140	6,32
150	4,65

Tabla 8: Tabla resumen de la potencia recibida con el demostrador tecnológico a diferentes distancias

Por tanto, queda demostrado el potencial de esta tecnología que pese a estar en una fase todavía inicial, puede resolver problemas que otras implementaciones de transferencia inalámbrica de potencia no consiguen. Un ejemplo es la tolerancia a la desalineación que existe

entre resonadores y que hacen que la eficiencia de transmisión se mantenga.

6.2 FUTURO DESARROLLO

Dado que los resonadores han sido caracterizados y el proceso de caracterización ha sido definido para que pueda ser replicable en el caso de construir nuevos resonadores, las propuestas de continuación del proyecto se centrarán básicamente en la mejora de la implementación electrónica del transmisor y del receptor.

Los puntos a seguir trabajando en el futuro pueden ser:

- Diseño de un set individual de conectorización: creación de una estructura fija que implemente conector y resonador de manera robusta y fiable. Mediante impresión 3D, se podría obtener estas estructuras personalizadas al diámetro y grosor de cada resonador.
- Mejora de la etapa transmisora: aumento de la potencia entregada a la carga mediante el diseño de un amplificador de potencia.
- Uso de métodos de *energy harvesting* en el receptor: utilización de etapas multiplicadores de tensión para conseguir tensiones útiles para la carga de baterías.

REFERENCIAS

[1] N. Tesla. System of transmission of electrical energy., March 20 1900. US Patent 645,576.

[2] S. Y. R. Hui, Wenxing Zhong, and C. K. Lee. A Critical Review of Recent Progress in Mid-Range Wireless Power Transfer. *IEEE Transactions on Power Electronics*, 29(9):4500–4511, sep 2014.

[3] Alanson P Sample, D A Meyer, and J R Smith. Analysis, Experimental Results, and Range Adaptation of Magnetically Coupled Resonators for Wireless Power Transfer. *IEEE Transactions on Industrial Electronics*, 58(2):544–554, feb 2011.

[4] Wireless Power Consortium. https://www.wirelesspowerconsortium.com/.

[5] André Kurs, Aristeidis Karalis, Robert Moffatt, J. D. Joannopoulos, Peter Fisher, and Marin Soljačić. *Science*, (5834):83–86.

[6] Mingzhao Song, Pavel Belov, and Polina Kapitanova. Wireless power transfer inspired by the modern trends in electromagnetics. *Applied Physics Reviews*, 4(2):021102, jun 2017.

[7] R.S. Kshetrimayum. A brief intro to metamaterials. *IEEE Potentials*, 23(5):44–46, jan 2005.

[8] Ana Diaz-Rubio, Jorge Carbonell, and Jose Sanchez-Dehesa. Wireless energy transfer between anisotropic metamaterials shells. *Ann. Physics*, 324:1783–1795, nov 2009.

[9] Hermann A. Haus. *Waves and fields in optoelectronics*. Prentice-Hall, 1984.

[10] RS Pro Silver 10 g Vial Epoxy Conductive Adhesive.

[11] Silver Conductive Epoxy, High Conductivity.

[12] Comsol, inc. https://www.comsol.com/.

[13] Thorlabs, inc. https://www.thorlabs.com/index.cfm.

[14] National instruments, labview. http://www.ni.com/es-es/shop/labview.html, jun 2017.

[15] Rohde & schwarz S.L. https://www.rohde-schwarz.com/us/product/zva-productstartpage_63493-9660.html, jun 2017.

[16] N5171b exg x-series rf analog signal generator, 9 khz to 6 ghz | keysight (formerly agilent's electronic measurement). http://www.keysight.com/en/pdx-x201905-pn-N5171B/exg-x-series-rf-analog-signal-generator-9-khz-to-6-ghz, jun 2017.

[17] Empower amplifier model: Sku 1109. http://www.empowerrf.com/products/display_amplifier.php?sku=1109, jun 2017.

[18] Mini-circuits. Jtos-400+. https://www.minicircuits.com/pdfs/JTOS-400+.pdf, jun 2017.

[19] Tqp7m9105 - qorvo. http://www.qorvo.com/products/p/TQP7M9105, jun 2017.

[20] Phihong. Psac12r-150. http://www.mouser.com/ds/2/323/PSAC12R-766223.pdf, jun 2017.

[21] Linear Technology. Basic concepts of linear regulator and switching mode power supplies. http://cds.linear.com/docs/en/application-note/AN140fa.pdf, jun 2017.

[22] Cincon. Ec3saw-2405. http://www.cincon.com/upload/media/data%20sheets/Data%20Sheet%20(DC)/SA%20CASE/EC3SAW-V15.pdf, jun 2017.

[23] Isola Group. Is 400. http://www.isola-group.com/wp-content/uploads/data-sheets/is400.pdf, jun 2017.

[24] Keysight. Msox3102t. http://www.keysight.com/en/pdx-x202188-pn-MSOX3102T/mixed-signal-oscilloscope-1-ghz-2-analog-plus-16-digital-channels, jun 2017.

Parte VII

ANEXO

A

ANEXO

A.1 RESULTADOS DE CARACTERIZACIÓN DE LOS RESONADORES

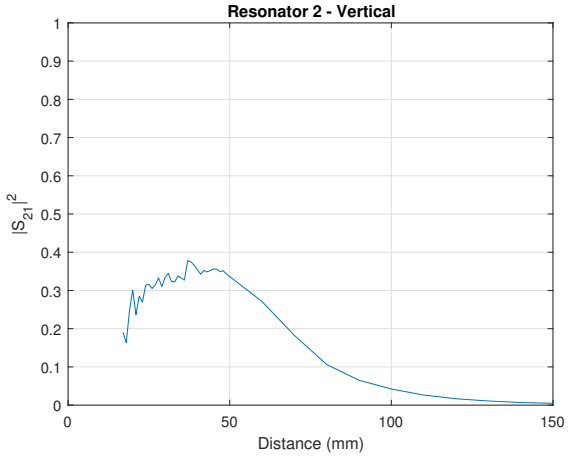

Figura 48: $|S_{21}|^2$ del resonador 2 en función de la distancia vertical. Ver tabla 1

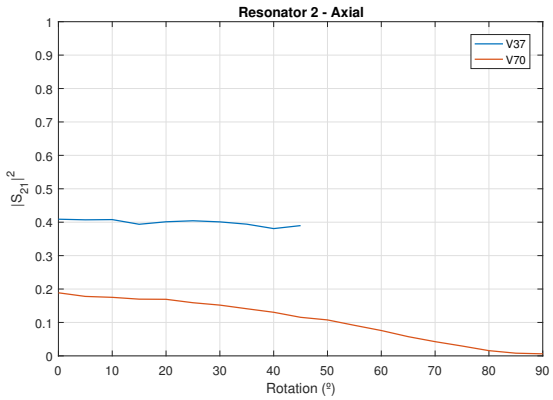

Figura 49: $|S_{21}|^2$ del resonador 2 en función del ángulo axial a 37 y 70 mm de distancia vertical. Ver tabla 1

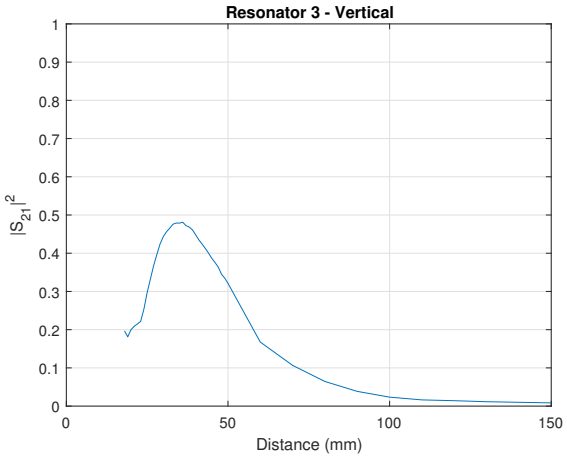

Figura 50: $|S_{21}|^2$ del resonador 3 en función de la distancia vertical. Ver tabla 1

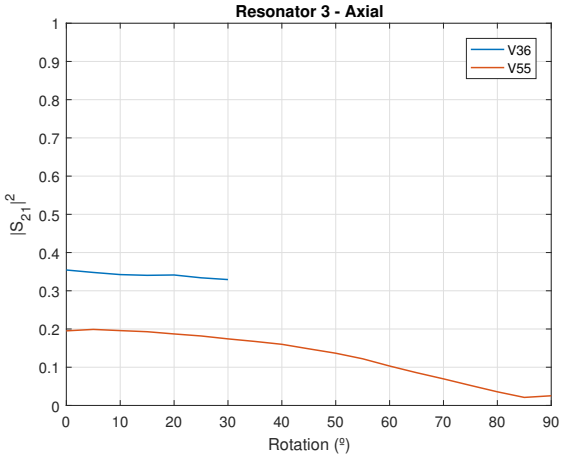

Figura 51: $|S_{21}|^2$ del resonador 3 en función del ángulo axial a 36 y 55 mm de distancia vertical. Ver tabla 1

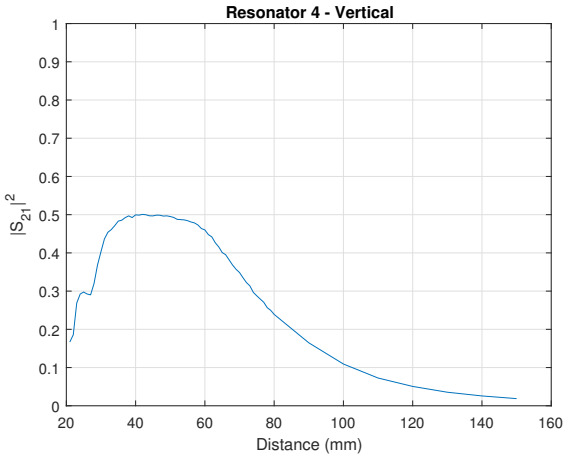

Figura 52: $|S_{21}|^2$ del resonador 4 en función de la distancia vertical

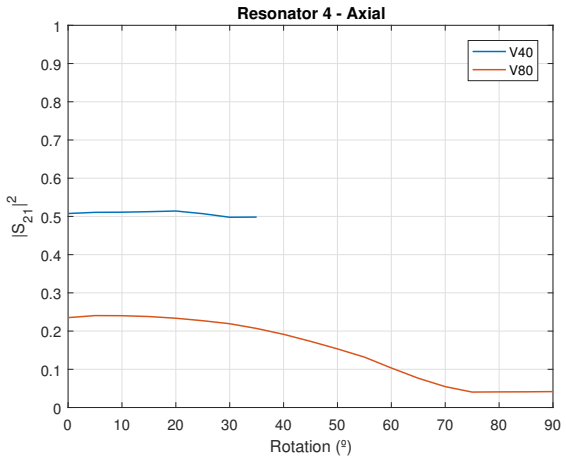

Figura 53: $|S_{21}|^2$ del resonador 4 en función del ángulo axial a 40 y 80 mm de distancia vertical

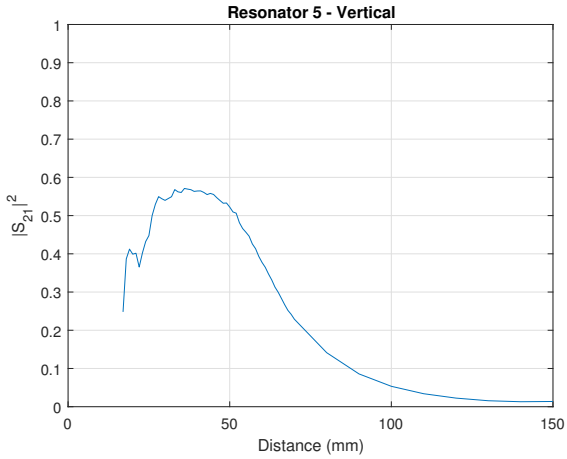

Figura 54: $|S_{21}|^2$ del resonador 5 en función de la distancia vertical. Ver tabla 1

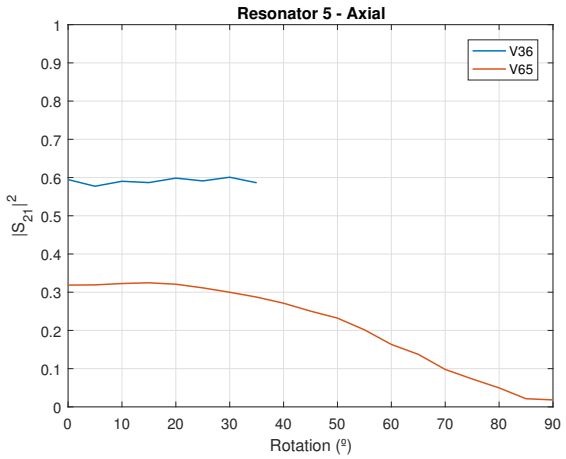

Figura 55: $|S_{21}|^2$ del resonador 4 en función del ángulo axial a 36 y 65 mm de distancia vertical. Ver tabla 1

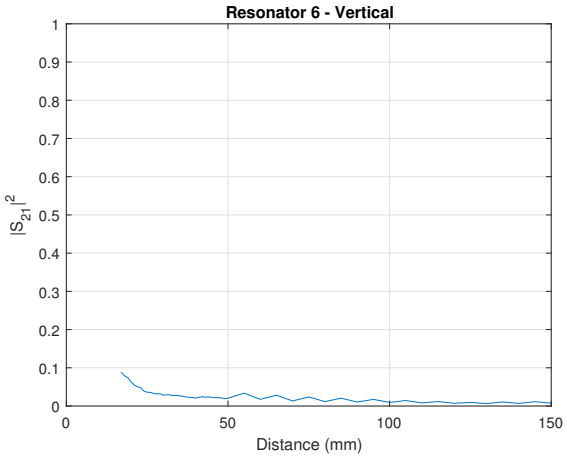

Figura 56: $|S_{21}|^2$ del resonador 6 en función de la distancia vertical. Ver tabla 1

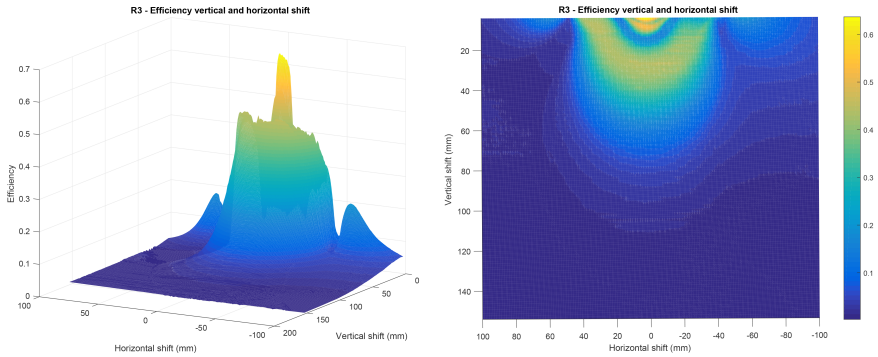

Figura 57: Mapa de la eficiencia del resonador 3 en función del desplazamiento vertical y horizontal. Ver tabla 1

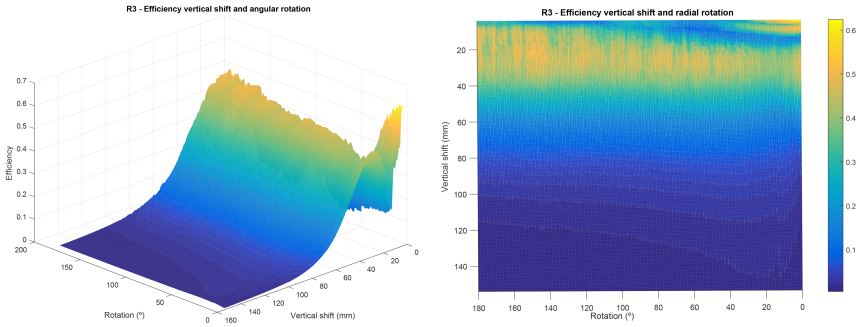

Figura 58: Mapa de la eficiencia del resonador 3 en función del desplazamiento vertical y angular. Ver tabla 1

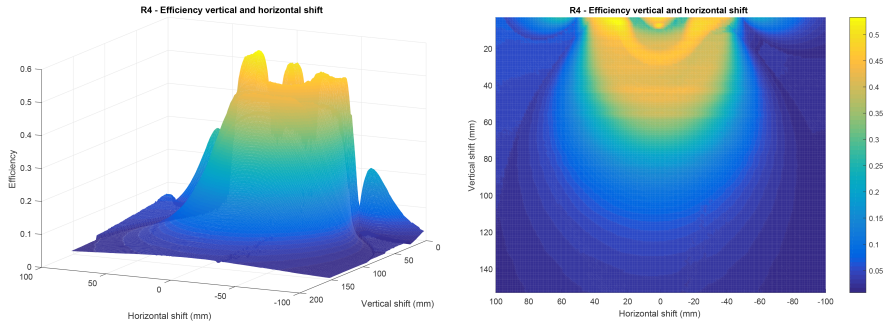

Figura 59: Mapa de la eficiencia del resonador 4 en función del desplazamiento vertical y horizontal. Ver tabla 1

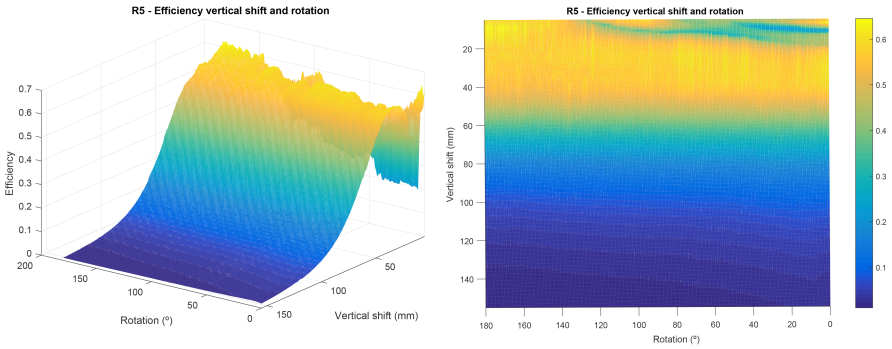

Figura 60: Mapa de la eficiencia del resonador 5 en función del desplazamiento vertical y angular. Ver tabla 1

A.2 DESGLOSE DE PEDIDOS

A.2.1 *Minicircuits*

Total 409.58 €.

Concepto	Unidades	Precio unidad	Total
JTOS-400+	20	16,4	328
TB-04	2	30,79	61,58
Gastos de envío	1	20	20

Tabla 9: Desglose pedido Minicircuits. Total 409.58 €

A.2.2 *RS Amidata*

A.2.2.1 *1 de julio de 2016*

Total 33.51 €.

Concepto	Unidades	Precio unidad	Total
Diode Schottky 15 mA 79 V DO35	50	0,225	11,25
Flyback Transformer Ui12V Uo200V 0.8W	1	5,25	5,25
Step-up transformer SMD	4	2,89	11,56
Guantes para hombre Hyflex, 1 par	1	5,45	5,45

Tabla 10: Desglose pedido RS Amidata de julio de 2016. Total 33.51 €

A.2.2.2 *23 de junio de 2017*

Total 39.01 €.

Concepto	Unidades	Precio unidad	Total
Adaptador de RF RS Pro, Conector Macho Recto SMA a Conector Macho SMA	3	9,08	32,96
Transporte		5	6,05

Tabla 11: Desglose pedido RS Amidata de junio de 2017. Total 39.01 €

A.2.3 *Eurocircuits*

A.2.3.1 *5 de septiembre de 2016*

Total 68.08 €.

A.2.3.2 *30 de marzo de 2017*

Total 69.93 €.

Concepto	Unidades	Precio unidad	Total
PCB 72x48 2 capas	5	12,88	64,4
Transporte		3,68	3,68

Tabla 12: Desglose pedido Eurocircuits julio 2016. Total 68.08 €

Concepto	Unidades	Precio unidad	Total
PCB 54x69 2 capas	5	13,25	66,25
Transporte		3,68	3,68

Tabla 13: Desglose pedido Eurocircuits marzo 2017. Total 69.93 €

A.2.4 *Mouser*

A.2.4.1 *22 de julio 2016*

Total 50.74 €.

Concepto	Unidades	Precio unidad	Total
Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 1uF 16volts X7R 10 %	10	0,107	1,07
Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 0.1uF 50volts X7R 10 %	10	0,042	0,42
Condensadores de cerámica multicapa (MLCC- SMD/SMT) 2.2PF 50V .1PF 0603	10	0,045	0,45
Inductores fijos 330nH 450mA 570MHz	5	0,215	1,075
Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0402 13pF 50Volts CoG 2 %	10	0,07	0,7
Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0.33uF 25volts X7R 10 %	5	0,161	0,805
Reguladores de voltaje lineal 5.0V 0.5A Positive	5	0,421	2,105
Amplificador de RF 50-1500MHz 1W 5V P1dB 30dBm @ 940MHz	3	11,92	35,76
Bloques terminales fijos 2P SIDE ENTRY 2.54mm	4	0,465	1,86
Inductores fijos 0603 12nH 5 %	15	0,14	2,1
Inductores fijos 0603 3.9nH .03nH	15	0,153	2,295
Inductores fijos 18 NH 5 % (R)	15	0,14	2,1

Tabla 14: Desglose pedido Mouser de julio de 2016. Total 50.74 €

A.2.4.2 *26 de agosto de 2016*

Total 54.18 €.

Concepto	Unidades	Precio unidad	Total
Heat Sinks Heatsink for TO-220 BLK ANODIZED	6	1,85	11,1
Heat Sinks HTSNK TO-220 218 247 BLK ANODIZED	6	2,03	12,18
Heat Sinks TO-220 LABOR SAVING	9	0,624	5,616
Linear Voltage Regulators Prec 500mA Regs 5V 0.5A Output	12	0,483	5,796
Fixed Inductors 18 NH 5 % (R)	45	0,14	6,3
Fixed Inductors 0603 3.9nH .03nH	45	0,153	6,885
Fixed Inductors 0603 12nH 5 %	45	0,14	6,3

Tabla 15: Desglose pedido Mouser de agosto de 2016. Total 54.18 €

A.2.4.3 *21 de marzo de 2017*

Total 89.3 €.

Concepto	Unidades	Precio unidad	Total
Multilayer Ceramic Capacitors MLCC - SMD/SMT 0603 33pF 250volts C0G 5 %	5	0,84	4,21
Fixed Inductors 0603 15nH 5 %	10	0,14	1,40
Multilayer Ceramic Capacitors MLCC - SMD/SMT 0603 18pF 100volt C0G +/-5 %	10	0,04	0,39
RF Amplifier 50-1500MHz 1W 5V P1dB 30dBm @ 940MHz	3	11,92	35,76
Fixed Inductors 0603 4.7nH .03nH	10	0,14	1,40
Isolated DC/DC Converters 3W, 9-36Vin, 5Vout 3W/Black Plastic/SIP	2	12,68	25,36
DC Power Connectors PCB 2.1MM	3	1,12	3,36
Standard LEDs - SMD RED WATER CLEAR	9	0,12	1,04
Thick Film Resistors - SMD 0603 2K20hm 5 % Anti Surge AEC-Q200	5	0,09	0,45
Wall Mount AC Adapters 12W 15V 0.8A Levl VI 2.1 No Blade	1	14,63	14,63
Aluminium Electrolytic Capacitors - Leaded 25volts 220uF 20 %	5	0,26	1,30

Tabla 16: Desglose pedido Mouser de marzo de 2017. Total 89.3 €

A.3 PROGRAMAS

Para el análisis de los datos tomados por LabView se ha realizado un *script* en MATLAB que interpreta los ficheros y obtiene una gráfica de eficiencia bidimensional. El código comentado se describe a continuación.

```
%% Extract info
% File name
fileID = fopen('R3 - rotation 1 degree.txt', 'r');
line = 0;
start = 'Start measurement';
preamble = {};
preamble_read = 0;

info_regexp = 'V(\d{1,3}) R(-?\d{1,3})';
horizontal_preamble = 'Vertical sweep: (\d{2,3}) cm. Vertical sweep points: (\d{2,3})';
vertical_preamble = 'Angular sweep: (\d{2,3})º. Angular sweep points: (\d{2,3})';
row = 0;
nPoints = 202;
while ~feof(fileID)
  line =fgetl(fileID);
  if(line == -1)
   break;
  end
% Read preamble
if(preamble_read == 0)
 % If end of preamble
 if(strcmp(line, start) == 1)
  preamble_read = 1;
  %%% Allocate indexTable, S11, S21, S12 and S22
  % Find tokens in horizontal line
  tokens = regexpi(preamble{2,1}, horizontal_preamble, 'tokens');
  % Get horizontal points
  hPoints = str2double(tokens{1,1}{1,2});
  % Find tokens in vertical line
  tokens = regexpi(preamble{3,1}, vertical_preamble, 'tokens');
  % Get vertical points
  vPoints = str2double(tokens{1,1}{1,2});
  % Allocate indexTable matrix
  indexTable = zeros( hPoints * vPoints , 1);
  % Allocate frequencies matrix
  frequencies = zeros( hPoints * vPoints, nPoints);
  % Allocate S11 matrix
  S11 = zeros( hPoints * vPoints, nPoints);
  % Allocate S21 matrix
```

```
  S21 = zeros( hPoints * vPoints, nPoints);
  % Allocate S12 matrix
  S12 = zeros( hPoints * vPoints, nPoints);
  % Allocate S22 matrix
  S22 = zeros( hPoints * vPoints, nPoints);
 else
  % If not end of preamble, keep storing
  preamble = [preamble; line];
 end
 % Read measurement
 else
   if(regexpi(line, info_regexp))
    tokens = regexpi(line, info_regexp, 'tokens');
    indexTable(counter,1) = str2double(tokens{1,1}{1,1});
    indexTable(counter,2) = str2double(tokens{1,1}{1,2});
    row = 0;
   else
    line = str2double(strsplit(line));
    if(row == 1)    % Frequency line
     frequencies(counter,:) = line;
    elseif(row == 2)    % S11 line
     S11(counter,:) = line;
    elseif(row == 3)    % S21 line
     S21(counter,:) = line;
    elseif(row == 4)    % S12 line
     S12(counter,:) = line;
    elseif(row == 5)    % S22 line
     S22(counter,:) = line;
    end %end if
   end %end else
   row = row + 1;   % Increase row
 end %end else
end %end while

%% Create mesh
maxS21 = max(S21,[],2);
S11AtS21Max = zeros(size(maxS21,1), size(maxS21,2));
maxS21Indexes = zeros(size(maxS21,1), size(maxS21,2));
frequencyAtS21 = zeros(size(maxS21,1), size(maxS21,2));
for i=1:length(S21) % Iterate S21 per row
 % Find column in the whole row where is the maximum value
 maxS21Indexes(i) = find(S21(i,:) == maxS21(i),1);
 % Find S11 corresponding to that S21 (same frequency)
 S11AtS21Max(i) = S11(i, find(S21(i,:) == maxS21(i),1));
 % Find frequencies at maxS21
 frequencyAtS21(i) = frequencies(i, find(S21(i,:) == maxS21(i),1));
```

```
end
% Get size of the mesh
meshSize = [length(unique(indexTable(:,2))) length(unique(indexTable(:,1)))];
% Initilize mesh for S21
maxS21grid = zeros(meshSize(2), meshSize(1));
% Initialize mesh for S11
S11AtS21Maxgrid = zeros(meshSize(2), meshSize(1));
% Initialize mesh for max S21 frequency
frequencyAtS21Grid =  zeros(meshSize(2), meshSize(1));
% Create mesh with unique values of X and Y
[X,Y] = meshgrid(unique(indexTable(:,2)), unique(indexTable(:,1)));

for i=1:length(maxS21)
 % Get the position of measure
 pos = indexTable(i,:);
 % Get y coordinate (vertical shift)
 y = pos(1);
 % Get x coordinate (angular rotation)
 x = pos(2);
 % Find row at which Y == vertical shift
 row = find(Y(:,1) == y);
 % Find column at which X == horizontal shift
 column = find(X(1,:) == x);
 % Save maxS21 at proper coordinate
 maxS21grid(row, column) = maxS21(i);
 %  Save S21 at proper coordinate
 S11AtS21Maxgrid(row, column) = S11AtS21Max(i);
 frequencyAtS21Grid(row, column) = frequencyAtS21(i);
end
%% Plot
figure
% maxS21
surf(X,Y,maxS21grid);
figure
% Efficiency |S_{21}|^2
surf(X,Y,abs(10.^(maxS21grid/20)).^2);
figure
% Efficiency |S_{21}|^2/(1-|S_{11}|^2)
surf(X,Y, abs(10.^(maxS21grid/20)).^2./(1-abs(10.^(S11AtS21Maxgrid/20)).^2));
xlabel('Rotation (º)');
ylabel('Vertical shift (mm)');
zlabel('Efficiency');
figure
surf(X,Y,frequencyAtS21Grid)
```

En caso de analizar un fichero de medidas vertical y radiales, hay que cambiar las primeras líneas 9, 10 y 11 del código anterior por las siguientes:

```
info_regexp = 'V(\d{1,3}) R(-?\d{1,3})';
horizontal_preamble = 'Vertical sweep: (\d{2,3}) cm. Vertical
    sweep points: (\d{2,3})';
vertical_preamble = 'Angular sweep: (\d{2,3})º. Angular
    sweep points: (\d{2,3})';
```

A.4 RESONATOR VIEWER

Para la visualización y manipulación de los datos extraídos de la caracterización manual de los resonadores se diseño un programa con interfaz gráfica para poder crear gráficas y exportar los datos en formato ASCII de manera sencilla. Este programa se ha realizado a través de MATLAB y utilizando la herramienta GUIDE. El aspecto visual de la aplicación es el que se muestra en la figura 61.

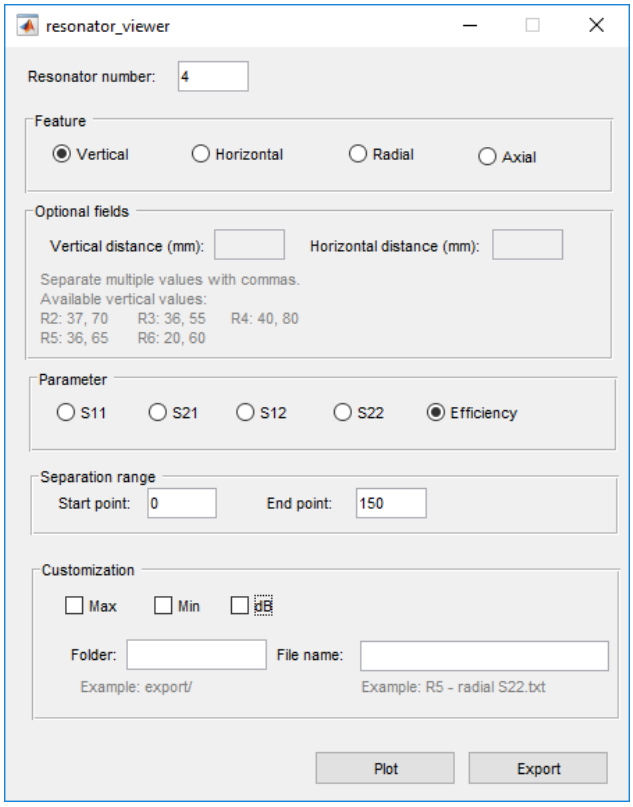

Figura 61: Interfaz gráfica de la aplicación Resonator Viewer

```
function varargout = resonator_viewer(varargin)
% RESONATOR_VIEWER MATLAB code for resonator_viewer.fig
```

```
3 %      RESONATOR_VIEWER, by itself, creates a new
  ↪ RESONATOR_VIEWER or raises the existing
4 %      singleton*.
5 %
6 %      H = RESONATOR_VIEWER returns the handle to a new
  ↪ RESONATOR_VIEWER or the handle to
7 %      the existing singleton*.
8 %
9 %
  ↪ RESONATOR_VIEWER('CALLBACK',hObject,eventData,handles,...)
  ↪ calls the local
10 %      function named CALLBACK in RESONATOR_VIEWER.M with
  ↪ the given input arguments.
11 %
12 %      RESONATOR_VIEWER('Property','Value',...) creates a
  ↪ new RESONATOR_VIEWER or raises the
13 %      existing singleton*.  Starting from the left,
  ↪ property value pairs are
14 %      applied to the GUI before
  ↪ resonator_viewer_OpeningFcn gets called.  An
15 %      unrecognized property name or invalid value makes
  ↪ property application
16 %      stop.  All inputs are passed to
  ↪ resonator_viewer_OpeningFcn via varargin.
17 %
18 %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
  ↪ allows only one
19 %      instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help
  ↪ resonator_viewer
24
25 % Last Modified by GUIDE v2.5 21-Apr-2017 12:31:22
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                    'gui_Singleton',  gui_Singleton, ...
31                    'gui_OpeningFcn',
                       ↪  @resonator_viewer_OpeningFcn, ...
32                    'gui_OutputFcn',
                       ↪  @resonator_viewer_OutputFcn, ...
33                    'gui_LayoutFcn',  [] , ...
34                    'gui_Callback',   []);
```

```
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State,
       ↪ varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46
47 % --- Executes just before resonator_viewer is made
   ↪ visible.
48 function resonator_viewer_OpeningFcn(hObject, eventdata,
   ↪ handles, varargin)
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata  reserved - to be defined in a future version
   ↪ of MATLAB
52 % handles    structure with handles and user data (see
   ↪ GUIDATA)
53 % varargin   command line arguments to resonator_viewer
   ↪ (see VARARGIN)
54
55 [handles.SParams, handles.filesTable] = extractSParams;
56
57 % Choose default command line output for resonator_viewer
58 handles.output = hObject;
59
60 % Update handles structure
61 guidata(hObject, handles);
62
63 % UIWAIT makes resonator_viewer wait for user response (see
   ↪ UIRESUME)
64 % uiwait(handles.figure1);
65
66
67 % --- Outputs from this function are returned to the
   ↪ command line.
68 function varargout = resonator_viewer_OutputFcn(hObject,
   ↪ eventdata, handles)
69 % varargout  cell array for returning output args (see
   ↪ VARARGOUT);
70 % hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function ResonatorValue_Callback(hObject, eventdata, handles)
% hObject    handle to ResonatorValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ResonatorValue as text
%        str2double(get(hObject,'String')) returns contents of ResonatorValue as a double


% --- Executes during object creation, after setting all properties.
function ResonatorValue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ResonatorValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in ExportButton.
function ExportButton_Callback(hObject, eventdata, handles)
% hObject    handle to ExportButton (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
resonator = str2double(get(handles.ResonatorValue, 'String'));
typeStr = get(get(handles.FeatureGroup, 'SelectedObject'), 'tag');
% if(strcmp(typeStr,'Vertical'))
%     type = 1;
% elseif(strcmp(typeStr, 'Horizontal'))
%     type = 2;
% else
%     type = 3;
% end

if(strcmp(typeStr,'Vertical'))
    type = 1;
elseif(strcmp(typeStr, 'Horizontal'))
    type = 2;
elseif(strcmp(typeStr, 'Radial'))
    type = 3;
elseif(strcmp(typeStr, 'Axial'))
    type = 5;
else
    type = 0;
end
parameter = get(get(handles.ParameterGroup, 'SelectedObject'), 'tag');
xlabelValue = 'Frequency (Hz)';
efficiencyEna = 0;
dBEna = get(handles.dBValue,'Value');
folder = get(handles.FolderValue, 'String');
fileName = get(handles.FileNameValue, 'String');

if(strcmp(parameter,'S11'))
    portA = 1;
    portB = 1;
elseif(strcmp(parameter, 'S21'))
    portA = 2;
    portB = 1;
elseif(strcmp(parameter, 'S12'))
    portA = 1;
    portB = 2;
elseif(strcmp(parameter, 'S22'))
    portA = 2;
```

```
145     portB = 2;
146 else
147     efficiencyEna = 1;
148 end
149 startPoint = str2double(get(handles.StartPointValue,
    ↪  'String'));
150 endPoint = str2double(get(handles.EndPointValue,
    ↪  'String'));
151
152 startFreq = str2double(get(handles.StartPointValue,
    ↪  'String'));
153 endFreq = str2double(get(handles.EndPointValue, 'String'));
154
155     verticalDistances = get(handles.VerticalDistanceValue,
        ↪  'String');
156     verticalDistances =
        ↪  str2double(strsplit(verticalDistances,','));
157
158     horizontalDistance =
        ↪  str2double(get(handles.HorizontalDistanceValue,
        ↪  'String'));
159
160 maxEna = get(handles.MaxValue,'Value');
161 minEna = get(handles.MinValue,'Value');
162
163 k = 0;
164 storedData = [];
165 for verticalDistance = verticalDistances
166     k = k+1;
167     if(efficiencyEna)
168         if(type == 1)
169             [x, outputData] = getEfficiency(type,
                ↪  handles.SParams, handles.filesTable,
                ↪  resonator, startPoint, endPoint, dBEna);
170         elseif(type == 2)
171             [x, outputData] = getEfficiency(type,
                ↪  handles.SParams, handles.filesTable,
                ↪  resonator, verticalDistance, startPoint,
                ↪  endPoint, dBEna);
172         elseif(type == 3 || type == 5)
173             [x, outputData] = getEfficiency(type,
                ↪  handles.SParams, handles.filesTable,
                ↪  resonator, verticalDistance,
                ↪  horizontalDistance, startPoint, endPoint,
                ↪  dBEna);
174         else
```

```
175         end
176     else
177         if(maxEna)
178             % plotSParams(type, sParams, filesTable,
                ↪  resonator, portA, portB, [verticalDistance,
                ↪  [horizontalDistance]], start, end, dB,
                ↪  title, xlabel, ylabel)
179             if(type == 1)
180                 [x, outputData] = getMaxS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB, startPoint,
                    ↪  endPoint, dBEna);
181             elseif(type == 2)
182                 [x, outputData] = getMaxS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB,
                    ↪  verticalDistance, startPoint, endPoint,
                    ↪  dBEna);
183             elseif(type == 3 || type == 5)
184                 [x, outputData] = getMaxS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB,
                    ↪  verticalDistance, horizontalDistance,
                    ↪  startPoint, endPoint, dBEna);
185             else
186             end
187         end
188         if(minEna)
189             if(type == 1)
190                 [x, outputData] = getMinS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB, startPoint,
                    ↪  endPoint, dBEna);
191             elseif(type == 2)
192                 [x, outputData] = getMinS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB,
                    ↪  verticalDistance, startPoint, endPoint,
                    ↪  dBEna);
193             elseif(type == 3 || type == 5)
194                 [x, outputData] = getMinS(type,
                    ↪  handles.SParams, handles.filesTable,
                    ↪  resonator, portA, portB,
                    ↪  verticalDistance, horizontalDistance,
                    ↪  startPoint, endPoint, dBEna);
195             else
```

```
196             end
197         end
198         if(~minEna && ~maxEna)
199             if(type == 1)
200                 [x, freq, outputData] = getSParams(type,
                    ↪ handles.SParams, handles.filesTable,...
201                     resonator, portA, portB, startPoint,
                        ↪ endPoint, dBEna);
202             elseif(type == 2)
203                 [x, freq, outputData] = getSParams(type,
                    ↪ handles.SParams, handles.filesTable,
                    ↪ ...
204                     resonator, portA, portB,
                        ↪ verticalDistance, startPoint,...
205                     endPoint, dBEna);
206             elseif(type == 3 || type == 5)
207                 [x, freq, outputData] = getSParams(type,
                    ↪ handles.SParams,
                    ↪ handles.filesTable,resonator, portA,
                    ↪ portB,
                    ↪ verticalDistance,horizontalDistance,
                    ↪ startPoint, endPoint, dBEna);
208             else
209             end
210         end
211     end
212     if(~minEna && ~maxEna && ~efficiencyEna)
213         storedData = outputData;
214     else
215         storedData(:,k) = outputData';
216     end
217 end
218 if(~minEna && ~maxEna && ~efficiencyEna)
219     if(type == 1)
220         saveAsciiTable(freq, x,  storedData, fileName,
            ↪ folder);
221     elseif(type == 2)
222         saveAsciiTable(freq, x, storedData,
            ↪ fileName,folder);
223     elseif(type == 3 || type == 5)
224         saveAsciiTable(freq, x, storedData, fileName,
            ↪ folder);
225     else
226     end
227 else
228     if(type == 1)
```

```
229         saveAsciiTable(x', 0, storedData(:,1),fileName,
            ↪  folder);
230     elseif(type == 2)
231         saveAsciiTable(x', verticalDistances',
            ↪  storedData,fileName, folder);
232     elseif(type == 3 || type == 5)
233         saveAsciiTable(x', verticalDistances', storedData,
            ↪  fileName, folder);
234     else
235     end
236 end
237 % --- Executes on button press in PlotButton.
238 function PlotButton_Callback(hObject, eventdata, handles)
239 % hObject    handle to PlotButton (see GCBO)
240 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
241 % handles    structure with handles and user data (see
    ↪  GUIDATA)
242 resonator = str2double(get(handles.ResonatorValue,
    ↪  'String'));
243 typeStr = get(get(handles.FeatureGroup, 'SelectedObject'),
    ↪  'tag');
244 if(strcmp(typeStr,'Vertical'))
245     type = 1;
246 elseif(strcmp(typeStr, 'Horizontal'))
247     type = 2;
248 elseif(strcmp(typeStr, 'Radial'))
249     type = 3;
250 elseif(strcmp(typeStr, 'Axial'))
251     type = 5;
252 else
253     type = 0;
254 end
255 parameter = get(get(handles.ParameterGroup,
    ↪  'SelectedObject'), 'tag');
256 xlabelValue = 'Frequency (Hz)';
257 efficiencyEna = 0;
258 dBEna = get(handles.dBValue,'Value');
259 if(dBEna)
260     ylabelTextdB = ' (dB)';
261 else
262     ylabelTextdB = '';
263 end
264 if(strcmp(parameter,'S11'))
265     portA = 1;
266     portB = 1;
```

```
267     ylabelValue = ['S' num2str(portA) num2str(portB)
        ↪  ylabelTextdB];
268 elseif(strcmp(parameter, 'S21'))
269     portA = 2;
270     portB = 1;
271     ylabelValue = ['S' num2str(portA) num2str(portB)
        ↪  ylabelTextdB];
272 elseif(strcmp(parameter, 'S12'))
273     portA = 1;
274     portB = 2;
275     ylabelValue = ['S' num2str(portA) num2str(portB)
        ↪  ylabelTextdB];
276 elseif(strcmp(parameter, 'S22'))
277     portA = 2;
278     portB = 2;
279     ylabelValue = ['S' num2str(portA) num2str(portB)
        ↪  ylabelTextdB];
280 else
281     efficiencyEna = 1;
282     if(type == 1 || type == 2)
283         xlabelValue = 'Distance (mm)';
284     else
285         xlabelValue = 'Rotation (º)';
286     end
287     ylabelValue = '|S_{21}|^2';
288 end
289 startPoint = str2double(get(handles.StartPointValue,
    ↪  'String'));
290 endPoint = str2double(get(handles.EndPointValue,
    ↪  'String'));
291 verticalDistances = get(handles.VerticalDistanceValue,
    ↪  'String');
292 verticalDistances =
    ↪  str2double(strsplit(verticalDistances,','));
293 horizontalDistance =
    ↪  str2double(get(handles.HorizontalDistanceValue,
    ↪  'String'));
294 maxEna = get(handles.MaxValue,'Value');
295 minEna = get(handles.MinValue,'Value');
296 fig = figure;
297 k = 0;
298 storedData = [];
299 for verticalDistance = verticalDistances
300     if(efficiencyEna)
301         if(type == 1)
```

```
302             plotEfficiency(type, handles.SParams,
                ↪  handles.filesTable, resonator, startPoint,
                ↪  endPoint, dBEna, ['Resonator '
                ↪  num2str(resonator) ' - ' typeStr],
                ↪  xlabelValue, ylabelValue, fig);
303         elseif(type == 2)
304             plotEfficiency(type, handles.SParams,
                ↪  handles.filesTable, resonator,
                ↪  verticalDistance, startPoint, endPoint,
                ↪  dBEna, ['Resonator ' num2str(resonator) ' -
                ↪  ' typeStr], xlabelValue, ylabelValue, fig);
305         elseif(type == 3 || type == 5)
306             plotEfficiency(type, handles.SParams,
                ↪  handles.filesTable, resonator,
                ↪  verticalDistance, horizontalDistance,
                ↪  startPoint, endPoint, dBEna, ['Resonator '
                ↪  num2str(resonator) ' - ' typeStr],
                ↪  xlabelValue, ylabelValue, fig);
                ↪  else
307         end
308     else
309         if(maxEna)
310             ylabelValue = ['max(S_{' num2str(portA)
                ↪  num2str(portB) '})' ylabelTextdB];
311             % plotSParams(type, sParams, filesTable,
                ↪  resonator, portA, portB, [verticalDistance,
                ↪  [horizontalDistance]], start, end, dB,
                ↪  title, xlabel, ylabel)
312             if(type == 1)
313                 xlabelValue = 'Distance (mm)';
314                 plotMaxS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, startPoint, endPoint, dBEna,
                    ↪  ['Resonator ' num2str(resonator) ' - '
                    ↪  typeStr], xlabelValue, ylabelValue,
                    ↪  fig);
315             elseif(type == 2)
316                 xlabelValue = 'Distance (mm)';
317                 plotMaxS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, verticalDistance, startPoint,
                    ↪  endPoint, dBEna, ['Resonator '
                    ↪  num2str(resonator) ' - ' typeStr],
                    ↪  xlabelValue, ylabelValue, fig);
318             elseif(type == 3 || type == 5)
319                 xlabelValue = 'Rotation (º)';
```

```
320                 plotMaxS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, verticalDistance,
                    ↪  horizontalDistance, startPoint,
                    ↪  endPoint, dBEna, ['Resonator '
                    ↪  num2str(resonator) ' - ' typeStr],
                    ↪  xlabelValue, ylabelValue, fig);
321             else
322             end
323         end
324         if(minEna)
325            ylabelValue = ['min(S_{' num2str(portA)
                ↪  num2str(portB) '})' ylabelTextdB];
326             % plotSParams(type, sParams, filesTable,
                ↪  resonator, portA, portB, [verticalDistance,
                ↪  [horizontalDistance]], start, end, dB,
                ↪  title, xlabel, ylabel)
327             if(type == 1)
328                 xlabelValue = 'Distance (mm)';
329                 plotMinS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, startPoint, endPoint, dBEna,
                    ↪  ['Resonator ' num2str(resonator) ' - '
                    ↪  typeStr], xlabelValue, ylabelValue,
                    ↪  fig);
330             elseif(type == 2)
331                 xlabelValue = 'Distance (mm)';
332                 plotMinS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, verticalDistance, startPoint,
                    ↪  endPoint, dBEna, ['Resonator '
                    ↪  num2str(resonator) ' - ' typeStr],
                    ↪  xlabelValue, ylabelValue, fig);
333             elseif(type == 3 || type == 5)
334                 xlabelValue = 'Rotation (º)';
335                 plotMinS(type, handles.SParams,
                    ↪  handles.filesTable, resonator, portA,
                    ↪  portB, verticalDistance,
                    ↪  horizontalDistance, startPoint,
                    ↪  endPoint, dBEna, ['Resonator '
                    ↪  num2str(resonator) ' - ' typeStr],
                    ↪  xlabelValue, ylabelValue, fig);
336             else
337             end
338         end
339         if(~minEna && ~maxEna)
```

```
340             xlabelValue = 'Frequency (Hz)';
341             if(type == 1)
342                 plotSParams(type, handles.SParams,
                    ↪ handles.filesTable, resonator, portA,
                    ↪ portB, startPoint, endPoint, dBEna,
                    ↪ ['Resonator ' num2str(resonator) ' - '
                    ↪ typeStr], xlabelValue, ylabelValue,
                    ↪ fig);
343             elseif(type == 2)
344                 plotSParams(type, handles.SParams,
                    ↪ handles.filesTable, resonator, portA,
                    ↪ portB, verticalDistance, startPoint,
                    ↪ endPoint, dBEna, ['Resonator '
                    ↪ num2str(resonator) ' - ' typeStr],
                    ↪ xlabelValue, ylabelValue, fig);
345             elseif(type == 3 || type == 5)
346                 plotSParams(type, handles.SParams,
                    ↪ handles.filesTable, resonator, portA,
                    ↪ portB, verticalDistance,
                    ↪ horizontalDistance, startPoint,
                    ↪ endPoint, dBEna, ['Resonator '
                    ↪ num2str(resonator) ' - ' typeStr],
                    ↪ xlabelValue, ylabelValue, fig);
347             else
348             end
349         end
350     end
351     hold on;
352 end
353
354
355 % --- Executes on button press in dBValue.
356 function dBValue_Callback(hObject, eventdata, handles)
357 % hObject    handle to dBValue (see GCBO)
358 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
359 % handles    structure with handles and user data (see
    ↪ GUIDATA)
360
361 % Hint: get(hObject,'Value') returns toggle state of
    ↪ dBValue
362
363
364
365 function FolderValue_Callback(hObject, eventdata, handles)
366 % hObject    handle to FolderValue (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FolderValue as text
%        str2double(get(hObject,'String')) returns contents of FolderValue as a double


% --- Executes during object creation, after setting all properties.
function FolderValue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FolderValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FileNameValue_Callback(hObject, eventdata, handles)
% hObject    handle to FileNameValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FileNameValue as text
%        str2double(get(hObject,'String')) returns contents of FileNameValue as a double


% --- Executes during object creation, after setting all properties.
```

```
398 function FileNameValue_CreateFcn(hObject, eventdata,
    ↪ handles)
399 % hObject    handle to FileNameValue (see GCBO)
400 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
401 % handles    empty - handles not created until after all
    ↪ CreateFcns called
402
403 % Hint: edit controls usually have a white background on
    ↪ Windows.
404 %       See ISPC and COMPUTER.
405 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
406     set(hObject,'BackgroundColor','white');
407 end
408
409
410
411 function StartPointValue_Callback(hObject, eventdata,
    ↪ handles)
412 % hObject    handle to StartPointValue (see GCBO)
413 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
414 % handles    structure with handles and user data (see
    ↪ GUIDATA)
415
416 % Hints: get(hObject,'String') returns contents of
    ↪ StartPointValue as text
417 %        str2double(get(hObject,'String')) returns contents
    ↪ of StartPointValue as a double
418
419
420 % --- Executes during object creation, after setting all
    ↪ properties.
421 function StartPointValue_CreateFcn(hObject, eventdata,
    ↪ handles)
422 % hObject    handle to StartPointValue (see GCBO)
423 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
424 % handles    empty - handles not created until after all
    ↪ CreateFcns called
425
426 % Hint: edit controls usually have a white background on
    ↪ Windows.
427 %       See ISPC and COMPUTER.
```

```
428 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
429     set(hObject,'BackgroundColor','white');
430 end
431
432
433
434 function EndPointValue_Callback(hObject, eventdata,
    ↪ handles)
435 % hObject    handle to EndPointValue (see GCBO)
436 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
437 % handles    structure with handles and user data (see
    ↪ GUIDATA)
438
439 % Hints: get(hObject,'String') returns contents of
    ↪ EndPointValue as text
440 %        str2double(get(hObject,'String')) returns contents
    ↪ of EndPointValue as a double
441
442
443 % --- Executes during object creation, after setting all
    ↪ properties.
444 function EndPointValue_CreateFcn(hObject, eventdata,
    ↪ handles)
445 % hObject    handle to EndPointValue (see GCBO)
446 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
447 % handles    empty - handles not created until after all
    ↪ CreateFcns called
448
449 % Hint: edit controls usually have a white background on
    ↪ Windows.
450 %       See ISPC and COMPUTER.
451 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
452     set(hObject,'BackgroundColor','white');
453 end
454
455
456
457 function HorizontalDistanceValue_Callback(hObject,
    ↪ eventdata, handles)
458 % hObject    handle to HorizontalDistanceValue (see GCBO)
459 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
```

```
460 % handles    structure with handles and user data (see
    ↪  GUIDATA)
461
462 % Hints: get(hObject,'String') returns contents of
    ↪  HorizontalDistanceValue as text
463 %        str2double(get(hObject,'String')) returns contents
    ↪  of HorizontalDistanceValue as a double
464
465
466 % --- Executes during object creation, after setting all
    ↪  properties.
467 function HorizontalDistanceValue_CreateFcn(hObject,
    ↪  eventdata, handles)
468 % hObject    handle to HorizontalDistanceValue (see GCBO)
469 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
470 % handles    empty - handles not created until after all
    ↪  CreateFcns called
471
472 % Hint: edit controls usually have a white background on
    ↪  Windows.
473 %       See ISPC and COMPUTER.
474 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪  get(0,'defaultUicontrolBackgroundColor'))
475     set(hObject,'BackgroundColor','white');
476 end
477
478
479
480 function VerticalDistanceValue_Callback(hObject, eventdata,
    ↪  handles)
481 % hObject    handle to VerticalDistanceValue (see GCBO)
482 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
483 % handles    structure with handles and user data (see
    ↪  GUIDATA)
484
485 % Hints: get(hObject,'String') returns contents of
    ↪  VerticalDistanceValue as text
486 %        str2double(get(hObject,'String')) returns contents
    ↪  of VerticalDistanceValue as a double
487
488
489 % --- Executes during object creation, after setting all
    ↪  properties.
```

```
490 function VerticalDistanceValue_CreateFcn(hObject, eventdata,
    ↪ handles)
491 % hObject    handle to VerticalDistanceValue (see GCBO)
492 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
493 % handles    empty - handles not created until after all
    ↪ CreateFcns called
494
495 % Hint: edit controls usually have a white background on
    ↪ Windows.
496 %       See ISPC and COMPUTER.
497 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
498     set(hObject,'BackgroundColor','white');
499 end
500
501
502 % --- Executes on button press in MaxValue.
503 function MaxValue_Callback(hObject, eventdata, handles)
504 % hObject    handle to MaxValue (see GCBO)
505 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
506 % handles    structure with handles and user data (see
    ↪ GUIDATA)
507
508 % Hint: get(hObject,'Value') returns toggle state of
    ↪ MaxValue
509
510
511 % --- Executes on button press in MinValue.
512 function MinValue_Callback(hObject, eventdata, handles)
513 % hObject    handle to MinValue (see GCBO)
514 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
515 % handles    structure with handles and user data (see
    ↪ GUIDATA)
516
517 % Hint: get(hObject,'Value') returns toggle state of
    ↪ MinValue
518
519
520 % --- Executes on button press in Vertical.
521 function Vertical_Callback(hObject, eventdata, handles)
522 % hObject    handle to Vertical (see GCBO)
523 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
```

```
524 % handles    structure with handles and user data (see
    ↪  GUIDATA)
525 handles.VerticalDistanceValue.Enable = 'off';
526 handles.HorizontalDistanceValue.Enable = 'off';
527 % Hint: get(hObject,'Value') returns toggle state of
    ↪  Vertical
528
529
530 % --- Executes on button press in Horizontal.
531 function Horizontal_Callback(hObject, eventdata, handles)
532 % hObject    handle to Horizontal (see GCBO)
533 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
534 % handles    structure with handles and user data (see
    ↪  GUIDATA)
535 handles.VerticalDistanceValue.Enable = 'on';
536 handles.HorizontalDistanceValue.Enable = 'off';
537 % Hint: get(hObject,'Value') returns toggle state of
    ↪  Horizontal
538
539
540 % --- Executes on button press in Radial.
541 function Radial_Callback(hObject, eventdata, handles)
542 % hObject    handle to Radial (see GCBO)
543 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
544 % handles    structure with handles and user data (see
    ↪  GUIDATA)
545 handles.VerticalDistanceValue.Enable = 'on';
546 handles.HorizontalDistanceValue.Enable = 'on';
547 % Hint: get(hObject,'Value') returns toggle state of Radial
548
549
550 % --- Executes on button press in Axial.
551 function Axial_Callback(hObject, eventdata, handles)
552 % hObject    handle to Axial (see GCBO)
553 % eventdata  reserved - to be defined in a future version
    ↪  of MATLAB
554 % handles    structure with handles and user data (see
    ↪  GUIDATA)
555 handles.VerticalDistanceValue.Enable = 'on';
556 handles.HorizontalDistanceValue.Enable = 'on';
557 % Hint: get(hObject,'Value') returns toggle state of Axial
558
559
560
```

```
561 function EndFrequencyValue_Callback(hObject, eventdata,
    ↪ handles)
562 % hObject    handle to EndFrequencyValue (see GCBO)
563 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
564 % handles    structure with handles and user data (see
    ↪ GUIDATA)
565
566 % Hints: get(hObject,'String') returns contents of
    ↪ EndFrequencyValue as text
567 %        str2double(get(hObject,'String')) returns contents
    ↪ of EndFrequencyValue as a double
568
569
570 % --- Executes during object creation, after setting all
    ↪ properties.
571 function EndFrequencyValue_CreateFcn(hObject, eventdata,
    ↪ handles)
572 % hObject    handle to EndFrequencyValue (see GCBO)
573 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
574 % handles    empty - handles not created until after all
    ↪ CreateFcns called
575
576 % Hint: edit controls usually have a white background on
    ↪ Windows.
577 %       See ISPC and COMPUTER.
578 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
579     set(hObject,'BackgroundColor','white');
580 end
581
582
583
584 function startFrequencyValue_Callback(hObject, eventdata,
    ↪ handles)
585 % hObject    handle to startFrequencyValue (see GCBO)
586 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
587 % handles    structure with handles and user data (see
    ↪ GUIDATA)
588
589 % Hints: get(hObject,'String') returns contents of
    ↪ startFrequencyValue as text
590 %        str2double(get(hObject,'String')) returns contents
    ↪ of startFrequencyValue as a double
```

```
591
592
593 % --- Executes during object creation, after setting all
    ↪ properties.
594 function startFrequencyValue_CreateFcn(hObject, eventdata,
    ↪ handles)
595 % hObject    handle to startFrequencyValue (see GCBO)
596 % eventdata  reserved - to be defined in a future version
    ↪ of MATLAB
597 % handles    empty - handles not created until after all
    ↪ CreateFcns called
598
599 % Hint: edit controls usually have a white background on
    ↪ Windows.
600 %       See ISPC and COMPUTER.
601 if ispc && isequal(get(hObject,'BackgroundColor'),
    ↪ get(0,'defaultUicontrolBackgroundColor'))
602     set(hObject,'BackgroundColor','white');
603 end
```

```
1  function [sParams, filesTable] = extractSParams()
2      vertical_regex = 'R(\d) - V(\d{2,3}).s2p';
3      hPerp_regex = 'R(\d) - H(\d{2,3}) V(\d{2,3}) PERP.s2p';
4      radial_regex = 'R(\d) - H(\d{2,3}) V(\d{2,3})
       ↪ Rad(\d{2,3}).s2p';
5      hPar_regex = 'R(\d) - H(\d{2,3}) V(\d{2,3}) PAR.s2p';
6      axial_regex = 'R(\d) - H(\d{2,3}) V(\d{2,3})
       ↪ A(\d{2,3}).s2p';
7
8      filesFolder = '';
9      unsortedFiles = dir([filesFolder '*.s2p']);
10
11     files = natsortfiles({unsortedFiles.name})';
12
13     %%% File table %%%
14     % length(files) rows x  5 columns
15     %| resonator | type |  V  |  H  | Angle |
16     %|     2     |   2  | 25  | 55  |  0    |
17     resonator = 1;
18     type = 2;
19     v = 3;
20     h = 4;
21     angle = 5;
22
23     %%% Type keys %%%
```

```
24      % Vertical: 1
25      % Horizontal perpendicular: 2
26      % Radial: 3
27      % Horizontal parallel: 4
28      % Axial: 5
29      vertical = 1;
30      hPerp = 2;
31      radial = 3;
32      hPar = 4;
33      axial = 5;
34
35      tableColumns = 5;
36      filesTable = zeros(length(files), tableColumns);
37
38      sParams = [];
39
40      for i=1:length(files)
41          validFile = 0;
42          if regexpi(char(files(i)), vertical_regex)
43              validFile = 1;
44              type = 1;
45              tokens = regexpi(char(files(i)), vertical_regex,
                ↪  'tokens');
46              filesTable(i,:) = [str2double(tokens{1,1}{1,1})
                ↪  type str2double(tokens{1,1}{1,2}) 0 0];
47          elseif(regexpi(char(files(i)), hPerp_regex))
48              validFile = 1;
49              type = 2;
50              tokens = regexpi(char(files(i)), hPerp_regex,
                ↪  'tokens');
51              filesTable(i,:) = [str2double(tokens{1,1}{1,1})
                ↪  type str2double(tokens{1,1}{1,3})
                ↪  str2double(tokens{1,1}{1,2}) 0];
52          elseif(regexpi(char(files(i)), radial_regex))
53              validFile = 1;
54              type = 3;
55              tokens = regexpi(char(files(i)), radial_regex,
                ↪  'tokens');
56              filesTable(i,:) = [str2double(tokens{1,1}{1,1})
                ↪  type str2double(tokens{1,1}{1,3})
                ↪  str2double(tokens{1,1}{1,2})
                ↪  str2double(tokens{1,1}{1,4})];
57          elseif(regexpi(char(files(i)), hPar_regex))
58              validFile = 1;
59              type = 4;
```

```
            tokens = regexpi(char(files(i)), hPar_regex,
            ↪ 'tokens');
            filesTable(i,:) = [str2num(tokens{1,1}{1,1})
            ↪ type str2num(tokens{1,1}{1,3})
            ↪ str2num(tokens{1,1}{1,2}) 0];
        elseif(regexpi(char(files(i)), axial_regex))
            validFile = 1;
            type = 5;
            tokens = regexpi(char(files(i)), axial_regex,
            ↪ 'tokens');
            filesTable(i,:) = [str2double(tokens{1,1}{1,1})
            ↪ type str2double(tokens{1,1}{1,3})
            ↪ str2double(tokens{1,1}{1,2})
            ↪ str2double(tokens{1,1}{1,4})];
        end
        if validFile
           sParams = [sParams;
            ↪ sparameters(char(strcat(filesFolder,
            ↪ char(files(i)))))];
        end
    end
end
```

```
% getEfficiency(type, sParams, filesTable, resonator,
↪ [verticalDistance [horizontalDistance]] startPoint,
↪ endPoint, dB)
function [x, outputData] = getEfficiency(varargin)
    if(nargin <= 0)
       error('Not enough input arguments.');
    end
    type = varargin{1};
    %(type, sParams, filesTable, resonator, startPoint,
    ↪ endPoint, dB)
    if(type == 1)
        SParams = varargin{2};
        filesTable = varargin{3};
        resonator = varargin{4};
        startPoint = varargin{5};
        endPoint = varargin{6};
        dB = varargin{7};
        [x, sParam] = getMaxS(type, SParams, filesTable,
        ↪ resonator, 2, 1, startPoint, endPoint, 0);
    % type, sParams, filesTable, resonator,
    ↪ verticalDistance, startPoint, endPoint, dB]
    elseif(type == 2)
```

```
        SParams = varargin{2};
        filesTable = varargin{3};
        resonator = varargin{4};
        verticalDistance = varargin{5};
        startPoint = varargin{6};
        endPoint = varargin{7};
        dB = varargin{8};
        [x, sParam] = getMaxS(type, SParams, filesTable,
        ↪  resonator, 2, 1, verticalDistance, startPoint,
        ↪  endPoint, 0);
    % type, sParams, filesTable, resonator,
    ↪  verticalDistance, horizontalDistance, startPoint,
    ↪  endPoint, dB] = varargin{1:end};
    elseif(type == 3 || type == 5)
        SParams = varargin{2};
        filesTable = varargin{3};
        resonator = varargin{4};
        verticalDistance = varargin{5};
        horizontalDistance = varargin{6};
        startPoint = varargin{7};
        endPoint = varargin{8};
        dB = varargin{9};
        [x, sParam] = getMaxS(type, SParams, filesTable,
        ↪  resonator, 2, 1, verticalDistance,
        ↪  horizontalDistance, startPoint, endPoint, 0);
    else
        error('Feature not configured. Allowed: 1 -
        ↪  vertical, 2 - horizontal_perp, 3 - radial');
    end
    outputData = (abs(sParam).^2)';
    if(dB)
        outputData = 10*log10(outputData);
    end
end
```

```
% getSParams(type, sParams, filesTable, resonator, portA,
↪  portB, start, end, dB)
function [x, outputSParams] = getMaxS(varargin)
    if(nargin <= 0)
       error('Not enough input arguments.');
    end
    % Query
    %     1       2          3          4        5      6
    ↪  7     8    9
```

```
8     %   type, s_params, filesTable, resonator, portA, portB,
      ↪  start, end, dB
9     if(varargin{1} == 1)
10        [type, sParams, filesTable, resonator, portA, portB,
          ↪  startPoint, endPoint, dB] = varargin{1:end};
11        rows = find(filesTable(:,1) == resonator &
          ↪  filesTable(:,2) == type & filesTable(:,3) >=
          ↪  startPoint & filesTable(:,3) <= endPoint);
12        x = filesTable(rows, 3)';
13    end
14    %     1       2          3          4        5      6
      ↪  7           8     9   10
15    %   type, s_params, filesTable, resonator, portA, portB,
      ↪  verticalDistance  start, end, dB
16    if(varargin{1} == 2)
17        [type, sParams, filesTable, resonator, portA, portB,
          ↪  verticalDistance, startPoint, endPoint, dB] =
          ↪  varargin{1:end};
18        rows = find(filesTable(:,1) == resonator &
          ↪  filesTable(:,2) == type & filesTable(:,3) ==
          ↪  verticalDistance &  filesTable(:,4) >=
          ↪  startPoint & filesTable(:,4) <= endPoint);
19        x = filesTable(rows, 4)';
20    end
21    %     1       2          3          4        5      6
      ↪  7               8               9     10   11
22    %   type, s_params, filesTable, resonator, portA, portB,
      ↪  verticalDistance, horizontalDistance,  start, end,
      ↪  dB
23    if(varargin{1} == 3 || varargin{1} == 5)
24        [type, sParams, filesTable, resonator, portA, portB,
          ↪  verticalDistance, horizontalDistance,
          ↪  startPoint, endPoint, dB] = varargin{1:end};
25        rows = find(filesTable(:,1) == resonator &
          ↪  filesTable(:,2) == type & filesTable(:,3) ==
          ↪  verticalDistance & filesTable(:,4) ==
          ↪  horizontalDistance & filesTable(:,5) >=
          ↪  startPoint & filesTable(:,5) <= endPoint);
26        x = filesTable(rows, 5)';
27    end
28    fetchedSParams = [];
29    for i=1:length(rows)
30        fetchedSParams = [fetchedSParams
          ↪  rfparam(sParams(rows(i)), portA, portB);];
31    end
32    outputSParams = zeros(1, size(fetchedSParams,2));
```

```
    for i=1:size(fetchedSParams,2)
        outputSParams(i) = max(fetchedSParams(:,i));
    end
    if(dB)
        outputSParams = 20*log10(abs(outputSParams));
    end
end
```

```
% getSParams(type, sParams, filesTable, resonator, portA,
  ↪  portB, start, end, dB)
function [x, outputSParams] = getMinS(varargin)
    if(nargin <= 0)
       error('Not enough input arguments.');
    end
    % Query
    %     1       2          3          4        5      6
     ↪  7     8    9
    %   type, s_params, filesTable, resonator, portA, portB,
     ↪  start, end, dB
    if(varargin{1} == 1)
        [type, sParams, filesTable, resonator, portA, portB,
         ↪  startPoint, endPoint, dB] = varargin{1:end};
        rows = find(filesTable(:,1) == resonator &
         ↪  filesTable(:,2) == type & filesTable(:,3) >=
         ↪  startPoint & filesTable(:,3) <= endPoint);
        x = filesTable(rows, 3)';
    end
    if(varargin{1} == 2)
        [type, sParams, filesTable, resonator, portA, portB,
         ↪  verticalDistance, startPoint, endPoint, dB] =
         ↪  varargin{1:end};
        rows = find(filesTable(:,1) == resonator &
         ↪  filesTable(:,2) == type & filesTable(:,3) ==
         ↪  verticalDistance &  filesTable(:,4) >=
         ↪  startPoint & filesTable(:,4) <= endPoint);
        x = filesTable(rows, 4)';
    end
    if(varargin{1} == 3 || varargin{1} == 5)
        [type, sParams, filesTable, resonator, portA, portB,
         ↪  verticalDistance, horizontalDistance,
         ↪  startPoint, endPoint, dB] = varargin{1:end};
```

```
        rows = find(filesTable(:,1) == resonator &
        ↪  filesTable(:,2) == type & filesTable(:,3) ==
        ↪  verticalDistance & filesTable(:,4) ==
        ↪  horizontalDistance & filesTable(:,5) >=
        ↪  startPoint & filesTable(:,5) <= endPoint);
        x = filesTable(rows, 5)';
    end
    fetchedSParams = [];
    for i=1:length(rows)
        fetchedSParams = [fetchedSParams
        ↪  rfparam(sParams(rows(i)), portA, portB);];
    end
    outputSParams = zeros(1, size(fetchedSParams,2));
    for i=1:size(fetchedSParams,2)
        outputSParams(i) = min(fetchedSParams(:,i));
    end
    if(dB)
        outputSParams = 20*log10(abs(outputSParams));
    end
end
```

```
% getSingleSParam(s_params, filesTable, type, resonator,
↪  portA, portB, [verticalDistance,
% horizontalDistance, radial], dB)
function [freq, sParam] = getSingleSParam(varargin)
    if(nargin <= 0)
        error('Not enough input arguments.');
    end
    % Vertical query
    %     1       2          3        4          5     6
    ↪  7
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  verticalDistance,
    %    8
    %   dB
    if(varargin{1} == 1)
        [type, sParams, filesTable, resonator, portA, portB,
        ↪  verticalDistance, dB] = varargin{1:end};
        row = find(filesTable(:,1) == resonator &
        ↪  filesTable(:,2) == type & filesTable(:,3) ==
        ↪  verticalDistance);

    % Horizontal perpendicular query
    %     1       2          3        4          5     6
    ↪  7
```

```
18      %   type, s_params, filesTable, resonator, portA, portB,
        ↪  horizontalDistance,
19      %       8           9
20      % verticalDistance, dB
21      elseif(varargin{1} == 2)
22          [type, sParams, filesTable, resonator, portA, portB,
            ↪  horizontalDistance, verticalDistance, dB] =
            ↪  varargin{1:end};
23          row = find(filesTable(:,1) == resonator &
            ↪  filesTable(:,2) == type & filesTable(:,3) ==
            ↪  verticalDistance & filesTable(:,4) ==
            ↪  horizontalDistance);
24      % Radial query
25      %     1       2          3        4          5     6
        ↪  7
26      %   type, s_params, filesTable, resonator, portA, portB,
        ↪  horizontalDistance,
27      %        8            9     10
28      % verticalDistance, radial, dB
29      elseif(varargin{1} == 3 || varargin{1} == 5)
30          [type, sParams, filesTable, resonator, portA, portB,
            ↪  horizontalDistance, verticalDistance, radial,
            ↪  dB] = varargin{1:end};
31          row = find(filesTable(:,1) == resonator &
            ↪  filesTable(:,2) == type & filesTable(:,3) ==
            ↪  verticalDistance & filesTable(:,4) ==
            ↪  horizontalDistance & filesTable(:,5) == radial
            ↪  );
32      % Horizontal perpendicular query
33      %     1       2          3        4          5     6
        ↪  7
34      %   type, s_params, filesTable, resonator, portA, portB,
        ↪  horizontalDistance,
35      %       8           9
36      % verticalDistance, dB
37      elseif(varargin{1} == 4)
38          [type, sParams, filesTable, resonator, portA, portB,
            ↪  horizontalDistance, verticalDistance, dB] =
            ↪  varargin{1:end};
39          row = find(filesTable(:,1) == resonator &
            ↪  filesTable(:,2) == type & filesTable(:,3) ==
            ↪  verticalDistance & filesTable(:,4) ==
            ↪  horizontalDistance);
40      else
41          error(['Feature type ' varargin{1} ' not valid.
            ↪  Check allowed range.']);
```

```
    end
    if(isempty(row))
        error('Distance not valid at getVerticalSParam.m
        ↪  line 2');
    end
    sParam = rfparam(sParams(row), portA, portB);
    freq = sParams(row).Frequencies;
    if(dB)
        sParam = 20*log10(abs(sParam));
    end
end
```

```
% getSParams(sParams, filesTable, type, resonator, portA,
↪  portB, start, end, dB)
function [x, freq, outputSParams] = getSParams(varargin)
    if(nargin <= 0)
       error('Not enough input arguments.');
    end
    % Query
    %     1       2          3          4        5      6
    ↪  7     8    9
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  start, end, dB
    if(varargin{1} == 1)
        [type, sParams, filesTable, resonator, portA, portB,
        ↪  startPoint, endPoint, dB] = varargin{1:end};
        rows = find(filesTable(:,1) == resonator &
        ↪  filesTable(:,2) == type & filesTable(:,3) >=
        ↪  startPoint & filesTable(:,3) <= endPoint);
        x = filesTable(rows, 3);
    end
    %  1       2       3              4       5      6
    ↪  7
    % type, sParams, filesTable, resonator, portA, portB,
    ↪  verticalDistance,
    %   8         9       10
    %startPoint, endPoint, dB
    if(varargin{1} == 2)
        [type, sParams, filesTable, resonator, portA, portB,
        ↪  verticalDistance, startPoint, endPoint, dB] =
        ↪  varargin{1:end};
        rows = find(filesTable(:,1) == resonator &
        ↪  filesTable(:,2) == type & filesTable(:,3) ==
        ↪  verticalDistance &  filesTable(:,4) >=
        ↪  startPoint & filesTable(:,4) <= endPoint);
```

```
        x = filesTable(rows, 4);
    end
    %  1       2       3              4       5       6
     ↪  7
    % type, sParams, filesTable, resonator, portA, portB,
     ↪  verticalDistance,
    %      8                 9         10      11
    % horzintalDistance, startPoint, endPoint, dB
    if(varargin{1} == 3 || varargin{1} == 5)
        [type, sParams, filesTable, resonator, portA, portB,
         ↪  verticalDistance, horizontalDistance,
         ↪  startPoint, endPoint, dB] = varargin{1:end};
        rows = find(filesTable(:,1) == resonator &
         ↪  filesTable(:,2) == type & filesTable(:,3) ==
         ↪  verticalDistance & filesTable(:,4) ==
         ↪  horizontalDistance & filesTable(:,5) >=
         ↪  startPoint & filesTable(:,5) <= endPoint);
        x = filesTable(rows, 5);
    end
    outputSParams = [];
    for i=1:length(rows)
        outputSParams = [outputSParams
         ↪  rfparam(sParams(rows(i)), portA, portB);];
    end
    outputSParams = abs(outputSParams);
    freq = sParams(rows(1)).Frequencies;
    if(dB)
        outputSParams = 20*log10(outputSParams);
    end
end
```

```
% plotEfficiency(type, sParams, filesTable, resonator,
 ↪  startPoint, endPoint, dB, title, xlabel, ylabel)
function fig = plotEfficiency(varargin)
type = varargin{1};
if(type == 1)
        [x, outputData] = getEfficiency(varargin{1:7});
        [titleText, xlabelText, ylabelText fig] =
         ↪  varargin{8:end};
    % type, sParams, filesTable, resonator,
     ↪  verticalDistance, startPoint, endPoint, dB]
    elseif(type == 2)
        [x, outputData] = getEfficiency(varargin{1:8});
        [titleText, xlabelText, ylabelText fig] =
         ↪  varargin{9:end};
```

```
    % type, sParams, filesTable, resonator,
    ↪  verticalDistance, horizontalDistance, startPoint,
    ↪  endPoint, dB] = varargin{1:end};
    elseif(type == 3 || type == 5)
        [x, outputData] = getEfficiency(varargin{1:9});
        [titleText, xlabelText, ylabelText fig] =
        ↪  varargin{10:end};
    else
        error('Feature not configured. Allowed: 1 -
        ↪  vertical, 2 - horizontal_perp, 3 - radial');
    end

    %[x, outputData] = getEfficiency(varargin{1:7});
    set(0, 'CurrentFigure', fig);
    fig;
    plot(x, outputData');
    grid on;
    title(titleText);
    xlabel(xlabelText);
    ylabel(ylabelText);
    ylim([0 1]);
end
```

```
% plotSParams(type, sParams, filesTable, resonator, portA,
↪  portB, [verticalDistance, [horizontalDistance]], start,
↪  end, dB, title, xlabel, ylabel)
function plotMaxS(varargin)
    type = varargin{1};
    %     1       2          3          4        5      6
    ↪  7     8    9   10
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  start, end, dB, fig
    if(type == 1)
        [x, outputSParams] = getMaxS(varargin{1:9});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{10:end};
    %     1       2          3          4        5      6
    ↪  7           8     9   10  11
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  verticalDistance  start, end, dB, fig
    elseif(type == 2)
        [x, outputSParams] = getMaxS(varargin{1:10});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{11:end};
```

```
    %     1       2          3          4        5      6
    ↪  7               8              9     10   11  12
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  verticalDistance, horizontalDistance,  start, end,
    ↪  dB, fig
    elseif(type == 3 || type == 5)
        [x, outputSParams] = getMaxS(varargin{1:11});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{12:end};
    else
    end
    set(0, 'CurrentFigure', fig);
    plot(x, outputSParams);
    grid on;
    title(titleText);
    xlabel(xlabelText);
    ylabel(ylabelText);
    %ylim([-40 0]);
end
```

```
% plotSParams(type, sParams, filesTable, resonator, portA,
↪  portB, [verticalDistance, [horizontalDistance]], start,
↪  end, dB, title, xlabel, ylabel)
function plotMinS(varargin)
    type = varargin{1};
    %     1       2          3          4        5      6
    ↪  7     8    9
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  start, end, dB
    if(type == 1)
        [x, outputSParams] = getMinS(varargin{1:9});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{10:end};
    %     1       2          3          4        5      6
    ↪  7           8     9   10
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  verticalDistance  start, end, dB
    elseif(type == 2)
        [x, outputSParams] = getMinS(varargin{1:10});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{11:end};
    %     1       2          3          4        5      6
    ↪  7               8              9     10   11
```

```
    %   type, s_params, filesTable, resonator, portA, portB,
    ↪  verticalDistance, horizontalDistance,  start, end,
    ↪  dB
    elseif(type == 3 || type == 5)
        [x, outputSParams] = getMinS(varargin{1:11});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{12:end};
    else
    end
    set(0, 'CurrentFigure', fig);
    plot(x, outputSParams);
    grid on;
    title(titleText);
    xlabel(xlabelText);
    ylabel(ylabelText);
    %ylim([-65 0]);
end
```

```
% plotSParams(type, sParams, filesTable, resonator, portA,
↪  portB, start, end, dB, title, xlabel, ylabel)
function plotSParams(varargin)
    type = varargin{1};
    if(type == 1)
        [x, freq, outputSParams] =
        ↪  getSParams(varargin{1:9});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{10:end};
    elseif(type == 2)
        [x, freq, outputSParams] =
        ↪  getSParams(varargin{1:10});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{11:end};
    elseif(type == 3 || type == 5)
        [x, freq, outputSParams] =
        ↪  getSParams(varargin{1:11});
        [titleText, xlabelText, ylabelText, fig] =
        ↪  varargin{12:end};
    else
    end
    % Vertical query
    if(varargin{1} == 1)
        preAppendText = 'Vertical: ';
        appendText = ' mm';
    % Horizontal perpendicular query
    elseif(varargin{1} == 2)
```

```
21          preAppendText = 'Horizontal: ';
22          appendText = ' mm';
23      % Radial query
24      elseif(varargin{1} == 3 || varargin{1} == 5)
25          preAppendText = 'Radial: ';
26          appendText = ' º';
27      % Horizontal perpendicular query
28      elseif(varargin{1} == 4)
29          preAppendText = 'Horizontal: ';
30          appendText = ' mm';
31      else
32          error(['Feature type ' varargin{1} ' not valid.
            ↪  Check allowed range.']);
33      end
34      legendValue = {};
35      for i=1:length(x)
36          legendValue = [legendValue ; [preAppendText
            ↪  num2str(x(i)) appendText]];
37      end
38      set(0, 'CurrentFigure', fig);
39      fig;
40      plot(freq, outputSParams);
41      grid on;
42      title(titleText);
43      xlabel(xlabelText);
44      ylabel(ylabelText);
45      legend(legendValue, 'Location', 'southeast');
46  end
```